Machine Learning
Bias-Variance Trade Off, KNN, GMM

Hosein Dadras
University of Tehran
Hoseindadras6@gmail.com
Hosiendadras@ut.ac.ir

Answer 1

1-1:

The bias-variance tradeoff is a fundamental concept that describes the tension between the error introduced by the model's approximations (bias) and the error introduced by the model's sensitivity to random fluctuations in the training set (variance). In the context of the Parzen window method, h_n represents the window width. A smaller h_n can lead to a model that captures more noise in the data (high variance) and a less smooth estimate, whereas a larger h_n can smooth out the noise but potentially miss important features (high bias).

In the k-nearest neighbors (KNN) method, k_n represents the number of nearest neighbors to consider. A smaller k_n makes the model more sensitive to noise (high variance), as it will be highly influenced by the nearest points. A larger k_n makes the model more robust to noise (lower variance) but can increase the bias if the neighborhood includes points from other classes or distributions.

1-2:

Proof of Divergence for $p(x)$

Given the function:

$$p(x) = \frac{K}{NV}\frac{1}{|x|}, \quad K > 1 \tag{1}$$

We evaluate the integral of $p(x)$ over the entire real line:

$$\int_{-\infty}^{+\infty} p(x)\,dx = \int_{-\infty}^{+\infty} \frac{K}{NV}\frac{1}{|x|}\,dx \tag{2}$$

$$= \frac{K}{NV}\left(\int_{-\infty}^{0} \frac{-1}{x}\,dx + \int_{0}^{+\infty} \frac{1}{x}\,dx\right) \tag{3}$$

$$= \frac{K}{NV}\left(\lim_{\epsilon\to 0^+}\int_{-\infty}^{-\epsilon} \frac{-1}{x}\,dx + \lim_{\epsilon\to 0^+}\int_{\epsilon}^{+\infty} \frac{1}{x}\,dx\right) \tag{4}$$

$$= \frac{K}{NV}\left(\lim_{\epsilon\to 0^+}\left[\ln|x|\right]_{-\infty}^{-\epsilon} + \lim_{\epsilon\to 0^+}\left[-\ln|x|\right]_{\epsilon}^{+\infty}\right) \tag{5}$$

$$= \frac{K}{NV}\left(\lim_{\epsilon\to 0^+}\ln(\epsilon) + \infty + \infty + \lim_{\epsilon\to 0^+}\ln(\epsilon)\right) \tag{6}$$

$$= \infty \tag{7}$$

Thus, $\int_{-\infty}^{+\infty} p(x)\,dx = \infty$, confirming that $p(x)$ is not a valid probability density function.

Answer 2-1:

Step-by-Step Calculation of the Parzen Window Estimator $\bar{p}_n(x)$

Given a uniform distribution $p(x) \sim U(0, a)$ and a Parzen window $\phi(x)$, we want to compute the Parzen window estimator $\bar{p}_n(x)$. We do this by finding the expected value of $\phi(x)$ with respect to $p(x)$.

Case 1: $x < 0$

For $x < 0$, we consider the definition of the Parzen window $\phi(x)$ which is given by:

$$\phi(x) = \begin{cases} +e^{-x} & \text{for } x > 0 \\ 0 & \text{for } x \leq 0 \end{cases}$$

Since x is less than 0, we are in the second part of the piecewise function where $\phi(x)$ is defined to be 0. Therefore, the estimation of the density using the Parzen window method, $\bar{p}_n(x)$, will also be 0, since the probability density of any point less than 0 is zero for a uniform distribution defined from 0 to a. Mathematically, this can be written as:

$$\bar{p}_n(x) = \int_{-\infty}^{\infty} p(u)\phi(x-u)\,du = 0$$

The integral evaluates to zero because for any u, $\phi(x-u)$ is zero due to the fact that $x-u$ is always less than zero if $x < 0$. Hence, $\bar{p}_n(x) = 0$ for $x < 0$.
When $x < 0$, v must be 0 too since $v < x$, and the integral is zero.

Case 2: $0 \leq x \leq a$

For $0 \leq x \leq a$, we must compute the estimator $\bar{p}_n(x)$ by integrating the product of the Parzen window $\phi(x)$ and the uniform distribution $p(x)$. Given the uniform distribution $p(x)$ is only non-zero within the interval $[0, a]$, the integral can be limited to this domain:

$$\bar{p}_n(x) = \int_{-\infty}^{\infty} p(u)\phi(x-u)\,du$$

However, because $p(x)$ is zero for $u < 0$ or $u > a$, and $\phi(x-u)$ is zero for $u > x$, we can further limit the integral's bounds from 0 to x:

$$\bar{p}_n(x) = \int_0^x \frac{1}{a}\phi(x-u)\,du$$

Substituting $\phi(x-u)$ with $e^{-(x-u)}$ for $x-u > 0$, which is guaranteed within our bounds, gives us:

$$\bar{p}_n(x) = \int_0^x \frac{1}{a}e^{-(x-u)}\,du$$

This simplifies to:

$$\bar{p}_n(x) = \frac{1}{a}\int_0^x e^{-x}e^{u}\,du$$

Since e^{-x} is a constant with respect to u, we can pull it out of the integral:

$$\bar{p}_n(x) = \frac{e^{-x}}{a}\int_0^x e^{u}\,du$$

Evaluating the integral of e^u from 0 to x gives us:

$$\bar{p}_n(x) = \frac{e^{-x}}{a}\left[e^u\right]_0^x = \frac{e^{-x}}{a}(e^x - 1)$$

Simplifying the expression, we end up with:

$$\bar{p}_n(x) = \frac{1}{a}(1 - e^{-x})$$

However, since x is scaled by the window width h_n in the Parzen estimator, we substitute x with $\frac{x}{h_n}$ to get the final form of the estimator for $0 \leq x \leq a$:

$$\bar{p}_n(x) = \frac{1}{a}(1 - e^{-\frac{x}{h_n}})$$

This concludes the derivation of $\bar{p}_n(x)$ for the case where x is between 0 and a inclusive. When $0 < x < a$, v can range from 0 to x. We obtain

$$\hat{p}(x) = \frac{1}{V_n}\int p(v)\phi\left(\frac{x-v}{h_n}\right)dv = \frac{1}{h_n}\int_0^x \frac{1}{a}\exp\left(\frac{v-x}{h_n}\right)dv$$

$$= \frac{1}{ah_n}\,h_n\exp\left(\frac{v-x}{h_n}\right)\Bigg|_{v=0}^{v=a} = \frac{1}{a}\left(1 - \exp\left(-\frac{x}{h_n}\right)\right).$$

Case 3: $x \geq a$

For $x \geq a$, we need to evaluate the estimator $\bar{p}_n(x)$ for values of x that are outside the interval of the uniform distribution. The integral remains the same, but we need to adjust the limits of integration to reflect that $\phi(x-u)$ is non-zero only when u is such that $x-u > 0$, i.e., $u < x$, and u is within the interval $[0, a]$ where $p(u)$ is non-zero:

$$\bar{p}_n(x) = \int_{-\infty}^{\infty} p(u)\phi(x-u)\,du$$

Considering the non-zero domains for $p(u)$ and $\phi(x-u)$, the bounds for u are from $x-a$ to a:

$$\bar{p}_n(x) = \int_{x-a}^{a} \frac{1}{a}\phi(x-u)\,du$$

Substitute $\phi(x-u)$ with $e^{-(x-u)}$ for the appropriate range and noting that e^{-x} is constant with respect to u, we have:

$$\bar{p}_n(x) = \frac{e^{-x}}{a}\int_{x-a}^{a} e^{u}\,du$$

The integral of e^u can be solved to give us the range of e^u from $x-a$ to a:

$$\bar{p}_n(x) = \frac{e^{-x}}{a}\left[e^{u}\right]_{x-a}^{a}$$

Evaluating this from $x-a$ to a yields:

$$\bar{p}_n(x) = \frac{e^{-x}}{a}(e^{a} - e^{x-a})$$

Simplify the expression by factoring out the common exponentials:

$$\bar{p}_n(x) = \frac{1}{a}(e^{a-x} - e^{-x})$$

And again, considering the scaling by h_n in the Parzen estimator, we substitute x with $\frac{x}{h_n}$ to get:

$$\bar{p}_n(x) = \frac{1}{a}\left(e^{\frac{a-x}{h_n}} - e^{-\frac{x}{h_n}}\right)$$

This simplifies to the final form of the estimator for $x \geq a$:

$$\bar{p}_n(x) = \frac{1}{a}\left(e^{\frac{a}{h_n}} - 1\right)e^{-\frac{x}{h_n}}$$

This concludes the derivation of $\bar{p}_n(x)$ for the case where x is greater than or equal to a. When $x \geq a$, v is not affected by x and ranges from 0 to a. We obtain

$$\hat{p}(x) = \frac{1}{V_n}\int p(v)\phi\left(\frac{x-v}{h_n}\right)dv = \frac{1}{h_n}\int_0^a \frac{1}{a}\exp\left(\frac{v-x}{h_n}\right)dv$$

$$= \frac{1}{ah_n}\,h_n \exp\left(\frac{v-x}{h_n}\right)\Bigg|_{v=0}^{v=a} = \frac{1}{a}\left(\exp\left(\frac{a}{h_n}\right) - 1\right)\exp\left(-\frac{x}{h_n}\right).$$

Answer 2-2:

The plot illustrates the Parzen window estimator $\bar{p}_n(x)$ as a function of x for $a = 1$ and three different values of the smoothing parameter h_n: 1, $\frac{1}{4}$, and $\frac{1}{16}$.
It shows the effect of the smoothing parameter on the estimator's behavior.
Smaller h_n values lead to a more sensitive estimator that captures more fluctuations in the data, whereas larger h_n values result in a smoother estimator.

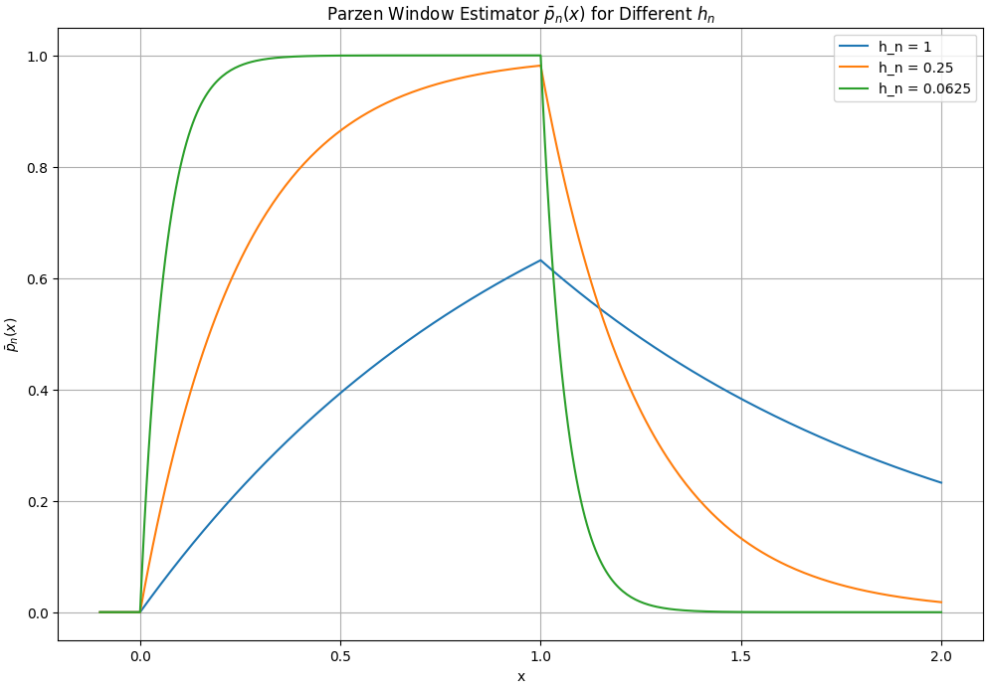

Figure 1: Parzen Window Estimator $\bar{p}_n(x)$ for $a = 1$ with different values of h_n.

The detailed implementation code for generating the plots is attached alongside this document.

Answer 2-3:

To assess the quality of the estimator $\hat{p}(x)$, one must consider the deviation of the estimate from the true value $p(x)$, commonly referred to as the bias. Specifically, the bias is defined as the expected difference between the estimator and the true probability density function, given by $E[\hat{p}(x)] - p(x)$.

To express this deviation relative to the true value, we introduce the relative bias, which is the absolute bias normalized by $p(x)$:

$$\text{relative bias}(x) = \frac{|\hat{p}(x) - p(x)|}{p(x)}.$$

Since $p(x)$ is uniform and equals $\frac{1}{a}$, the relative bias simplifies to:

$$\text{relative bias}(x) = |1 - a\hat{p}(x)| = e^{-\frac{x}{h_n}},$$

where the last equality holds due to the specific form of $\hat{p}(x)$ for the considered estimator.

The estimator's bias diminishes as x increases within the interval $(0, a)$. To ensure the bias remains within acceptable bounds, say less than 1% for the majority (99%) of the distribution range, we set a requirement on the relative bias at $x = \frac{a}{100}$:

$$\text{relative bias}\left(\frac{a}{100}\right) < 0.01.$$

This condition is equivalent to:

$$\exp\left(-\frac{a}{100h_n}\right) < 0.01.$$

Solving the above inequality for h_n allows us to find the necessary bandwidth h_n that meets this criterion. The solution yields:

$$h_n = \frac{a}{100 \ln 100},$$

providing a quantitative measure for the parameter h_n to ensure that the estimator $\hat{p}(x)$ maintains a bias below the specified threshold across the defined interval.

Answer 2-4:

Given the condition that the relative bias must be less than 1% for 99% of the interval $0 < x < a$, we are tasked to find the parameter h_n when $a = 1$. The bias is quantified as

$$\text{bias}(x) = \left|\frac{p(x) - \hat{p}(x)}{p(x)}\right|,$$

where $p(x)$ is the true probability density function and $\hat{p}(x)$ is the Parzen-window estimate.

By setting the bias at $x = \frac{a}{100}$ to 0.01, we have the equation

$$\exp\left(-\frac{a}{100h_n}\right) = 0.01.$$

Solving this equation for h_n yields

$$h_n = \frac{a}{100 \ln(100)} \approx 0.0022.$$

With this value of h_n, we can plot the Parzen-window estimate $\hat{p}_n(x)$ over the range $0 \leq x \leq 0.05$. The resulting plot demonstrates the steep increase in the estimated probability density as x approaches 0.01, at which point the relative bias is approximately 0.01, confirming that $\hat{p}(0.01) = 0.99$.

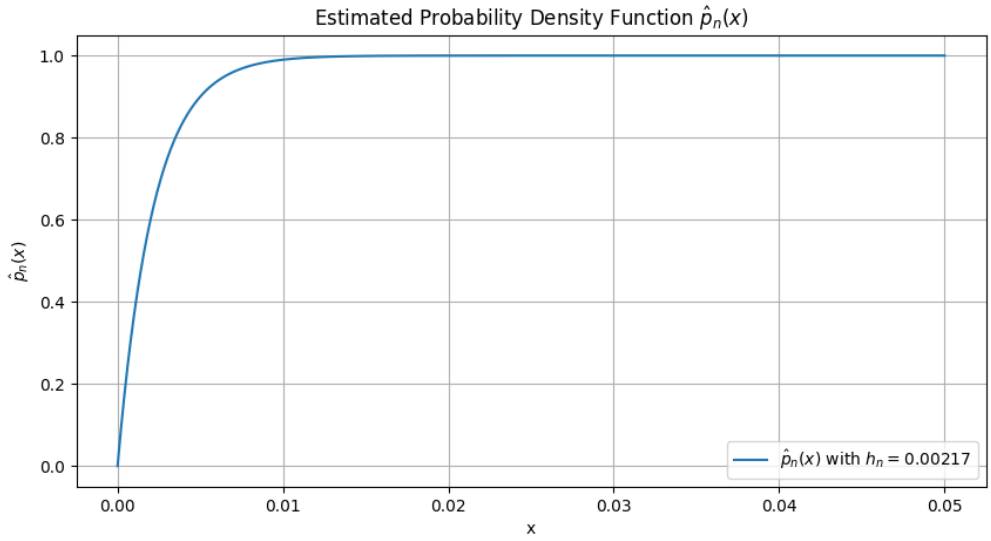

Figure 2: The Parzen-window estimate $\hat{p}_n(x)$ plotted for $0 \leq x \leq 0.05$ with $h_n \approx 0.0022$, demonstrating the required bias condition is met at $x = 0.01$.

Answer 3:

We'll prove $\hat{p}_n(x) \sim \mathcal{N}(\mu, \sigma^2 + h_n^2)$ by direct computation. An alternative approach would be to use the Fourier convolution theorem, which states that convolution becomes pointwise multiplication in the Fourier basis. We start with

$$\hat{p}_n(x) = \int \frac{1}{h_n} \phi\left(\frac{x-v}{h_n}\right) p(v)\, dv,$$

and if $\phi(x) \sim \mathcal{N}(0,1)$ then it is easy to see that $\frac{1}{h_n}\phi\left(\frac{x}{h_n}\right) \sim \mathcal{N}(0, h_n^2)$. We'll expand the exponents, write it as a quadratic function of u, integrate out the u-variable and using algebra transform the result into a more well-known form. The integral is

$$\frac{1}{\sqrt{2\pi\sigma^2}\sqrt{2\pi h_n^2}} \int \exp\left[-\frac{1}{2}\left(\frac{(x-v)^2}{h_n^2} + \frac{(v-\mu)^2}{\sigma^2}\right)\right] dv,$$

and we wish to integrate out the u. To do so, we write the exponent as a function of u. Defining $\beta = x^2\sigma^2 + h_n^2\mu^2$, the exponent may be written as

$$\frac{\sigma^2(x^2 - 2xv + v^2) + h_n^2(v^2 - 2v\mu + \mu^2)}{(h_n\sigma)^2} = \frac{v^2(\sigma^2 + h_n^2) - 2v(x\sigma^2 + \mu h_n^2) + \beta}{(h_n\sigma)^2}.$$

Now we introduce $y = u\sqrt{\sigma^2 + h_n^2}$, since we are aiming to write the exponent as $(y-a)^2/b^2 + c$ for some y. Defining a and completing the square, the right hand side of the equation above may be written as

$$\frac{y^2 - 2v\left(\frac{x\sigma^2 + \mu h_n^2}{\sqrt{\sigma^2 + h_n^2}}\right) + \beta}{(h_n\sigma)^2} = \frac{(y-\alpha)^2 - \alpha^2 + \beta}{(h_n\sigma)^2}.$$

We return to the integral in Equation (4), use $du = (\sigma^2 + h_n^2)^{-1/2} dy$ and write

$$\frac{1}{\sqrt{2\pi\sigma^2}\sqrt{2\pi h_n^2}\sqrt{\sigma^2 + h_n^2}} \exp\left[-\frac{1}{2}\left(\frac{-\alpha^2 + \beta}{(h_n\sigma)^2}\right)\right] \int \exp\left[-\frac{1}{2}\left(\frac{y-\alpha}{h_n\sigma}\right)^2\right] dy,$$

where the integral is evaluated easily since it's merely $\mathcal{N}(\alpha, h_n^2\sigma^2)$. We have

$$\hat{p}_n(x) = \frac{1}{\sqrt{2\pi}\sqrt{\sigma^2 + h_n^2}} \exp\left[-\frac{1}{2}\left(\frac{\beta - \alpha^2}{(h_n\sigma)^2}\right)\right].$$

$$\frac{-\alpha^2 + \beta}{(h_n\alpha)^2} = \left[\frac{1}{(h_n\alpha)^2}\left[\frac{(\alpha^2 + h_n^2)(x^2\sigma^2 + h_n^2\mu^2)}{\sigma^2 + h_n^2} - \frac{(x\sigma^2 + \mu h_n^2)^2}{\sigma^2 + h_n^2}\right]\right]$$

$$\hat{p}_n(x) = \frac{1}{\sqrt{2\pi}\sqrt{\sigma^2 + h_n^2}} \exp\left[-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2 + h_n^2}\right]$$

Answer 4-1:

Let us consider the EM algorithm for maximizing the likelihood $L(\theta)$. The following inequality gives a lower bound on $L(\theta)$ for any set of distributions:

$$\sum_{i=1}^{m} p(x^i, \theta) = \sum_i \log \sum_z p(x^i, z^i; \theta) \geq \sum_i \sum_z Q(z^i) \log \frac{p(x^i, z^i; \theta)}{Q(z^i)}$$

By Jensen's Inequality, we have that for any concave function f,

$$f\left(\mathbb{E}_{z^i \sim Q}\left[\frac{p(x^i, z^i; \theta)}{Q(z^i)}\right]\right) \geq \mathbb{E}_{z^i \sim Q}\left[f\left(\frac{p(x^i, z^i; \theta)}{Q(z^i)}\right)\right]$$

where the expectations $\mathbb{E}$ are with respect to z_i from Q_i. Now, we define $Q_i(z^i)$ as follows:

$$Q_i(z^i) = \frac{p(x^i, z^i; \theta)}{\sum_z p(x^i, z^i; \theta)} = \frac{p(x^i, z^i; \theta)}{p(x^i; \theta)} = p(z^i | x^i; \theta)$$

For each i, we repeat the E-step:

$$Q_i(z^i) = p(z^i | x^i; \theta)$$

And the M-step is:

$$\theta = \underset{\theta}{\text{argmax}} \sum_i \sum_{z^i} Q_i(z^i) \log \frac{p(x^i, z^i; \theta)}{Q_i(z^i)}$$

Answer 4-2:

Given the likelihood function $L(\theta)$, we have:

$$\begin{aligned} L(\theta) &= p(x, z|\theta)p(\theta) \\ \ln(L(\theta)) &= \ln(P(\theta)) + \ln(p(x, z|\theta)) \\ \ln p(x, z|\theta) &= \ln p(x|z, \theta) + \ln p(z, \theta) \end{aligned}$$

For different cases of x, we have: **Case 1:** $x = b$

$$\ln p(x = b|z, \theta) + \ln p(z|\theta) = n_b \ln \left(\frac{1}{3}(1 - \theta)\right)$$

Case 2: $x = d$

$$\ln p(x = d|z, \theta) = n_d \ln \left(\frac{1}{3}(1 - \theta)\right)$$

Case 3: $x = a$ or $x = c$

$$\begin{aligned} &\ln p(x = a \text{ or } c|z = a, \theta) + \ln p(z|\theta) \\ &Q(z = a|x, \theta) = \frac{p(z = a|\theta)p(x = a \text{ or } c|z, \theta)}{1/2\theta^{\hat{\theta}}} \\ &\frac{1}{1 + 2\theta^{\hat{\theta}}}(n_a + n_c)(1 + \ln(1/3)) \\ &x, a \text{ or } c + 6 \rightarrow (z = c|x, a)(\ln p(x = a \text{ or } c)) \\ &z = c, \theta) \ln p(z = c|\theta) \\ &2\theta^{\hat{\theta}}(n_a + n_c) \ln \left(\frac{2}{3}\theta\right) \end{aligned}$$

Then, the logarithm of the likelihood function is:

$$\begin{aligned} \ln(L(\theta)) = \ln &\left(\frac{\Gamma(v_1 + v_2)}{\Gamma(v_1)\Gamma(v_2)}\right) \\ &+ \frac{1}{2} + 2\theta^{\hat{\theta}}((n_a + n_c) + \ln(1/3)) \\ &+ \frac{2\theta^{\hat{\theta}}}{1 + 2\theta^{\hat{\theta}}}((n - (n_b + n_d)) \ln \left(\frac{2}{3}\theta\right)) \end{aligned}$$

For the prior distribution of θ:

$$\begin{aligned} \ln(p(\theta)) &= \ln \left(\frac{\Gamma(v_1, v_2)}{\Gamma(v_1)\Gamma(v_2)}\theta^{v_1 - 1}(1 - \theta)^{v_2 - 1}\right) \\ &= \ln(y) + (v_1 - 1) \ln \theta + (v_2 - 1) \ln(1 - \theta) \end{aligned}$$

Answer 5:

For each player i, the probability of winning w_t out of m_t possible games on day t is modeled using a binomial distribution. The binomial distribution gives the exact probability of a number of successes in m_t independent trials, each with success probability p_i:

$$P(W_t = w_t|p_i, m_t) = \binom{m_t}{w_t} p_i^{w_t}(1-p_i)^{m_t-w_t} \tag{8}$$

Let m_s and w_s be the parameters of a binomial distribution, the equations can be written as: For I:

$$I = \binom{m_s}{w_s} p_i^{w_s}(1-p_i)^{m_s-w_s}$$

For II:

$$II = \binom{m_s}{w_s} p_i^{w_s}(1-p_s)^{m_s-w_s} c_i$$

where $\binom{m_t}{w_t}$ is the number of ways w_t successes can occur in m_t trials.
Our mixture model assumes that for each day t, a player is selected from the k players with probability $c_t[i]$ and the observed w_t victories come from this player. The probability of observing w_t victories on day t is given by the following mixture distribution:

$$P(W_t = w_t|m_t) = \sum_{i=1}^{k} c_t[i] \cdot P(W_t = w_t|p_i, m_t) \tag{9}$$

This equation is a mixture of binomial distributions where each is attributed to a specific player and weighted by the probability of selecting that player $c_t[i]$.
These relationships can be used within the framework of an Expectation-Maximization (EM) algorithm to estimate the parameters p_i and c_t. During the Expectation step (E-step), we compute the posterior probability $Q_t^{(i)}[k]$ for each player k and each day t. During the Maximization step (M-step), we use these posterior probabilities to update our estimates of p_i and c_t. This process continues until the parameter estimates converge.
In the E-step of the EM algorithm, we update the posterior probabilities $Q_t^{(i)}[k]$, which represent the probability that player k was the one playing on day t. These probabilities are updated based on the observed wins w_t and the number of games played m_t, as well as the current parameter estimates:

$$Q_t^{(i)}[k] = \frac{c_t[k] \cdot P(W_t = w_t|p_k, m_t)}{\sum_{j=1}^{K} c_t[j] \cdot P(W_t = w_t|p_j, m_t)} \tag{10}$$

where:
- $c_t[k]$ is the probability of selecting player k for playing on day t before seeing the outcomes.
- $P(W_t = w_t|p_k, m_t)$ is the probability of observing w_t wins out of m_t games for player k with win probability p_k, calculated using the binomial distribution. - The denominator normalizes the weighted probabilities to ensure that they sum to 1 across all players for each day t.
The values of $Q_t^{(i)}[k]$ are then used in the subsequent M-step to update the estimates for the probabilities p_k and c_t to maximize the likelihood of the observed data.

The parameter $\pi[k]$ is updated during the M-step of the Expectation-Maximization algorithm. Based on the posterior probabilities $Q_t[k]$ computed in the E-step, the update formula for $\pi[k]$ is given by:

$$\pi[k] = \frac{1}{n} \sum_{t=1}^{n} Q_t[k] \tag{11}$$

where:

- n is the total number of observations or days.
- $Q_t[k]$ is the posterior probability that player k was selected on day t.

This equation represents the average probability of player k being selected across all days. For Q_{Ji}:

$$Q_{Ji} = \frac{\binom{m_s}{w_s} p_i^{w_s} (1-p_i)^{m_s - w_s}}{\sum_{i=1}^{k} \binom{m_s}{w_s} p_i^{w_s} (1-p_s)^{m_s - w_s} c_i}$$

For the loglikelihood:

$$\text{loglikelihood} = \prod_{i=1}^{k} \prod_{J=1}^{k} p_J^{w_i} \binom{m_i}{w_i} (1-p_i)^{m_i - w_i} c_i$$

For the sum of logs:

$$\sum_{i=1}^{k} \sum_{j=1}^{k} \left(\ln \binom{m_i}{w_i} + w_i \ln(p_j) + \binom{m_i}{w_i} \ln(c_i) \right)$$

Let L be defined as:

$$L = Q_{iJ} \times \text{loglikelihood}$$

And finally:

$$\frac{\binom{m_i}{w_i} p_i^{w_i} (1-p_i)^{m_i - w_i} c_i}{\sum_{i=1}^{n} \binom{m_i}{w_i} p_J^{w_i} (1-p_i)^{m_i - w_i} c_i}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \left(\ln \binom{m_i}{w_i} + w_i \ln(p_j) + \binom{m_i}{w_i} \ln(1-p_i) + \ln(c_i) + \lambda \left(\sum_{i=1}^{n} c_i \right) \right)$$

Answer 6:

Introduction

The K-Nearest Neighbors (KNN) algorithm is a simple, non-parametric method used for classification and regression. In this report, we apply the KNN algorithm to classify species in the Iris dataset, which consists of 150 samples from three species with four features each: sepal length, sepal width, petal length, and petal width.

Methodology

The dataset was split into training (75%) and evaluation (25%) sets. A custom KNN classifier was implemented in Python, utilizing the NumPy library for efficient computation. We experimented with different values of k to find the optimal number of neighbors for classification.

Results

Iris Dataset Loaded and General Information

The Iris dataset is a classic in the field of machine learning and statistics. It contains data from 150 Iris flowers, consisting of the following general information:

- Number of Classes: 3 (Iris Setosa, Iris Versicolor, Iris Virginica)
- Number of Samples: 150
- Number of Features: 4
  - Sepal Length (cm)
  - Sepal Width (cm)
  - Petal Length (cm)
  - Petal Width (cm)
- Data Format: Floating-point numbers (float64)
- Target Format: Integer labels (int64) corresponding to the class
- Feature Names: Listed as feature names
- Target Names: Array of target names as class labels

This dataset is widely used for classification tasks, particularly for teaching purposes due to its simplicity and clear distinction among classes.

Dataset Scatter Plot

The scatter plot provides a visual representation of the dataset with respect to two features: sepal length and width.

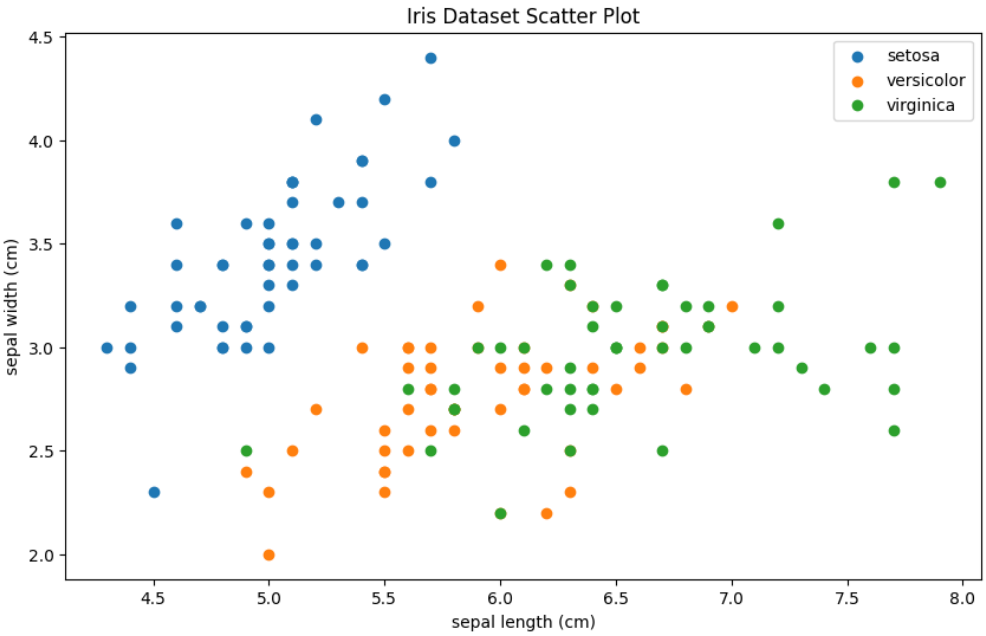

Figure 3: Scatter plot of the Iris dataset.

Accuracy Evaluation

The KNN classifier's accuracy was evaluated on the training and evaluation sets for k values ranging from 1 to 10.

1 Model Evaluation for $k = 5$

Upon training the KNN classifier with $k = 5$, the following accuracies were observed:

- Training Accuracy: 96.43%
- Evaluation Accuracy: 100%

These results indicate that the classifier is highly effective on the given Iris dataset. While the training accuracy shows that the model has learned well from the training data, the perfect score on the evaluation set is noteworthy. However, such an ideal outcome may also hint at the model's overfitting tendencies, which is a common issue in machine learning tasks with small datasets. A further evaluation using a method like k-fold cross-validation could provide a more comprehensive understanding of the model's predictive performance and its ability to generalize beyond the given data.

Visual Representation of Model Accuracy

The accuracy of the model can be visualized in the plot below, which contrasts the actual classes with the predicted classes obtained from the KNN classifier for the evaluation data.

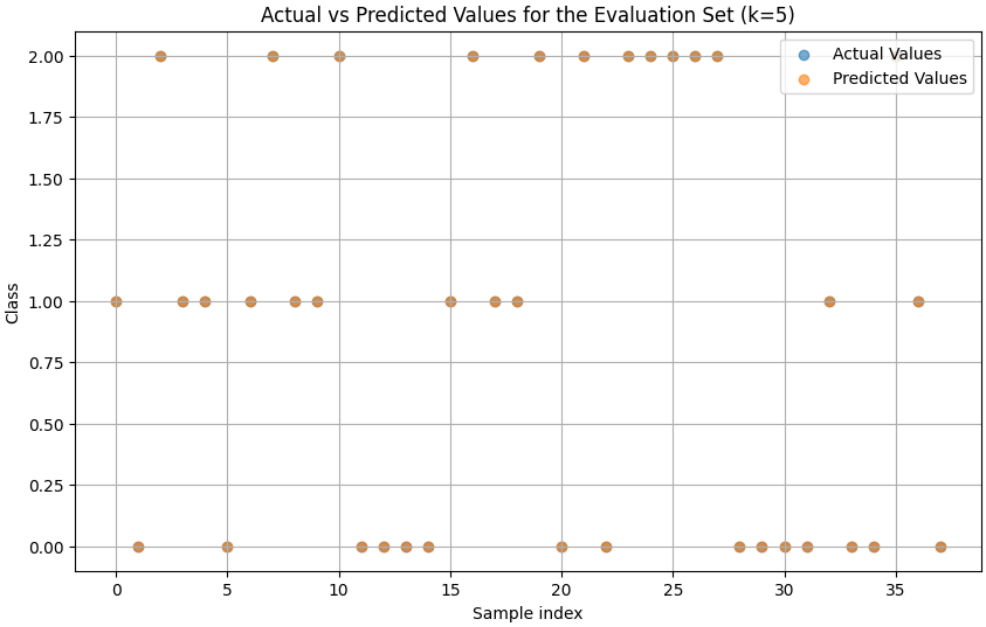

Figure 4: Actual vs. Predicted class labels on the evaluation set for the KNN classifier with $k = 5$.

As seen in the figure, the predictions align perfectly with the actual class labels, which is reflected in the 100% accuracy on the evaluation set.

Accuracy Evaluation Across Different k Values

The KNN classifier was trained and evaluated with different numbers of neighbors, k, ranging from 1 to 10. The accuracy of the classifier on both the training and evaluation datasets was recorded for each k value to determine the optimal configuration.

Optimal Number of Neighbors

The optimal number of neighbors was found to be $k = 1$, which provided a perfect accuracy of 100% on the evaluation dataset. This result is highlighted in the plot below with a vertical dashed line.

Analysis and Discussion

While a k value of 1 may generally lead to a less robust model due to overfitting, in the case of the Iris dataset, it seems to be sufficient for accurate classification. The high accuracy suggests that the dataset's classes are linearly separable to some extent. However, reliance on the closest neighbor alone is typically not advisable for more complex or overlapping datasets.

Accuracy Plot

The following figure shows the training and test accuracies for the range of k values considered. The best k is indicated, allowing for a direct visual comparison of how the choice of k influences model performance.

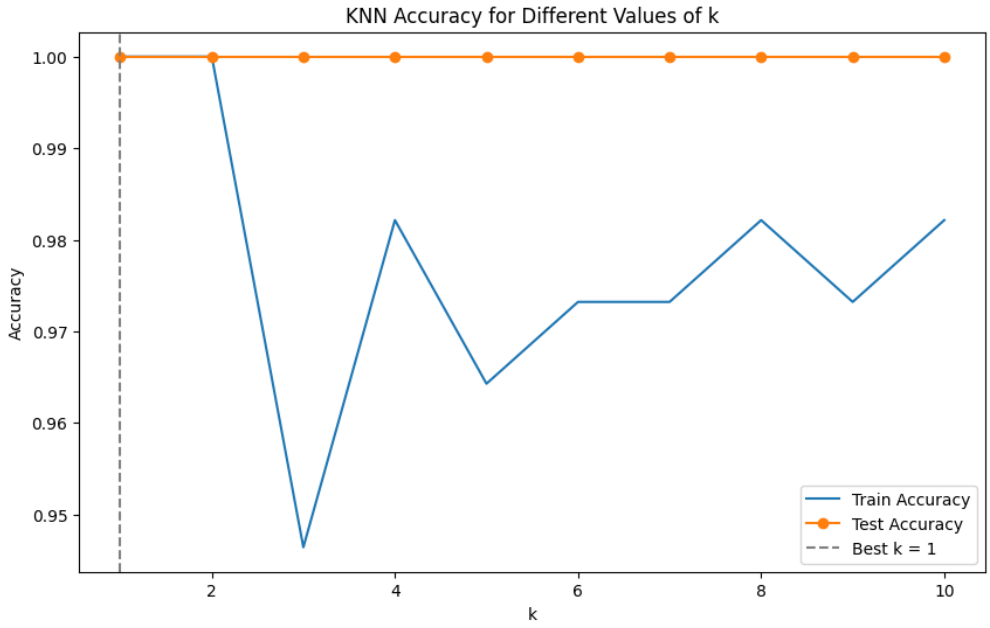

Figure 5: Training and test accuracies for different values of k in the KNN classifier. The best k value is marked with a dashed line.

Given these results, further investigation with cross-validation would be beneficial to validate the findings and ensure that the model is not overfitted to the specific characteristics of the training set.

Answer 7:

Introduction

This report presents an analysis of a generated dataset using Kernel Density Estimation (KDE) with the Parzen window method. The dataset is generated and analyzed under different bandwidths to understand the impact of the Parzen window size on the estimated distribution.

Data Generation

The dataset, denoted as X, is generated using a mixture of two normal distributions. The code snippet used for generation is as follows:

```
N = 1000
np.random.seed(1)
X = np.concatenate((np.random.normal(0, 1, int(0.3 * N)),
                    np.random.normal(5, 1, int(0.7 * N))))[:, np.newaxis]
```

The resulting dataset X has 1000 samples, with the first few samples shown below:

$$X = \begin{bmatrix} 1.62434536 \\ -0.61175641 \\ -0.52817175 \\ \vdots \\ -0.24937038 \end{bmatrix}$$

Kernel Density Estimation with Optimal Bandwidth

KDE with a Gaussian kernel is applied to estimate the distribution of X. A grid search method is used to determine the optimal bandwidth, which is found to be approximately 0.379. The KDE plot with this optimal bandwidth is as follows:

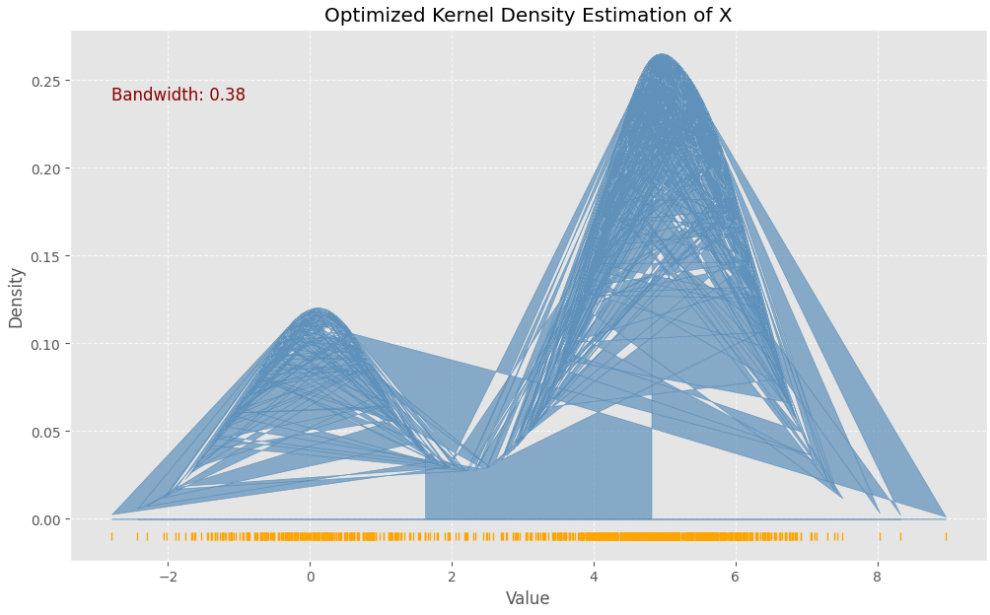

Figure 6: Kernel Density Estimation of X with Optimal Bandwidth

Effect of Parzen Window Size on Estimated Distribution

The impact of different Parzen window sizes (bandwidths) on the estimated distribution is investigated. Three bandwidths - 10, 1, and 0.1 - are chosen for this analysis. The plot below illustrates the effect of these bandwidths on the KDE:

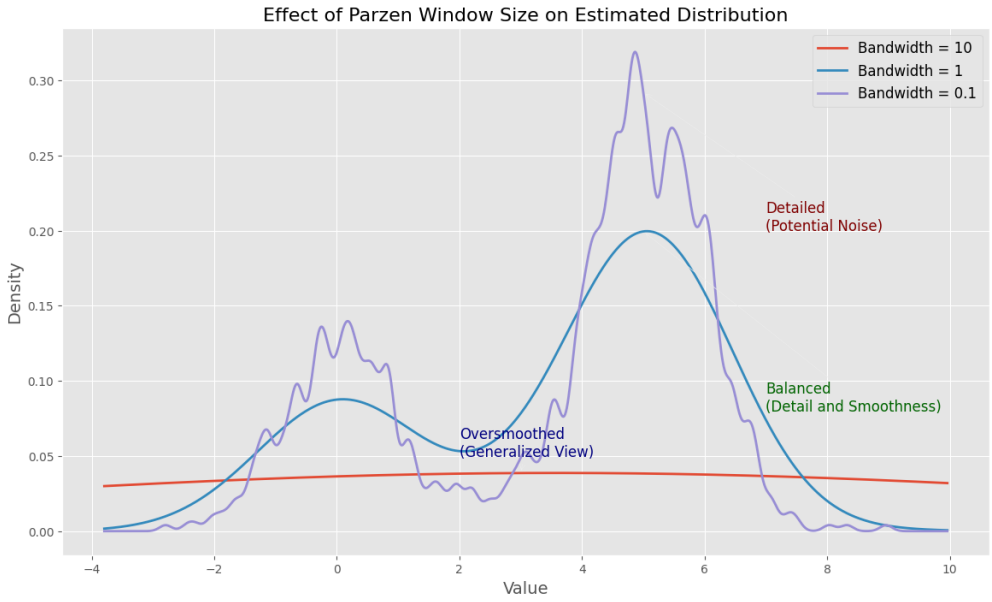

Figure 7: Effect of Parzen Window Size on Estimated Distribution

Observations

- **Bandwidth 10**: Results in an oversmoothed distribution, losing specific details and providing a generalized view.
- **Bandwidth 1**: Offers a balanced view, capturing both detail and smoothness effectively.
- **Bandwidth 0.1**: Provides a detailed view but may introduce noise, potentially overfitting the sample data.

The analysis demonstrates the significant impact of the Parzen window size on the estimated distribution of data using KDE. Choosing an appropriate bandwidth is crucial for obtaining a reliable estimation of the data distribution.

Answer 8:

The first step in our analysis was to create a synthetic dataset known as "Noisy Moons". This dataset is often used to demonstrate the behavior of clustering algorithms. We generated the dataset using the following code snippet provided in the exercise:

```
from sklearn import cluster, datasets, mixture
noisy_moons= datasets.make_moons(n_samples=500, noise=0.11)
```

Scatter Diagram of the Dataset

The scatter diagram below (Figure 8) shows the created "Noisy Moons" dataset. Each point in the scatter plot represents a data sample, with two distinct classes that resemble two interleaving half-circles with added noise.

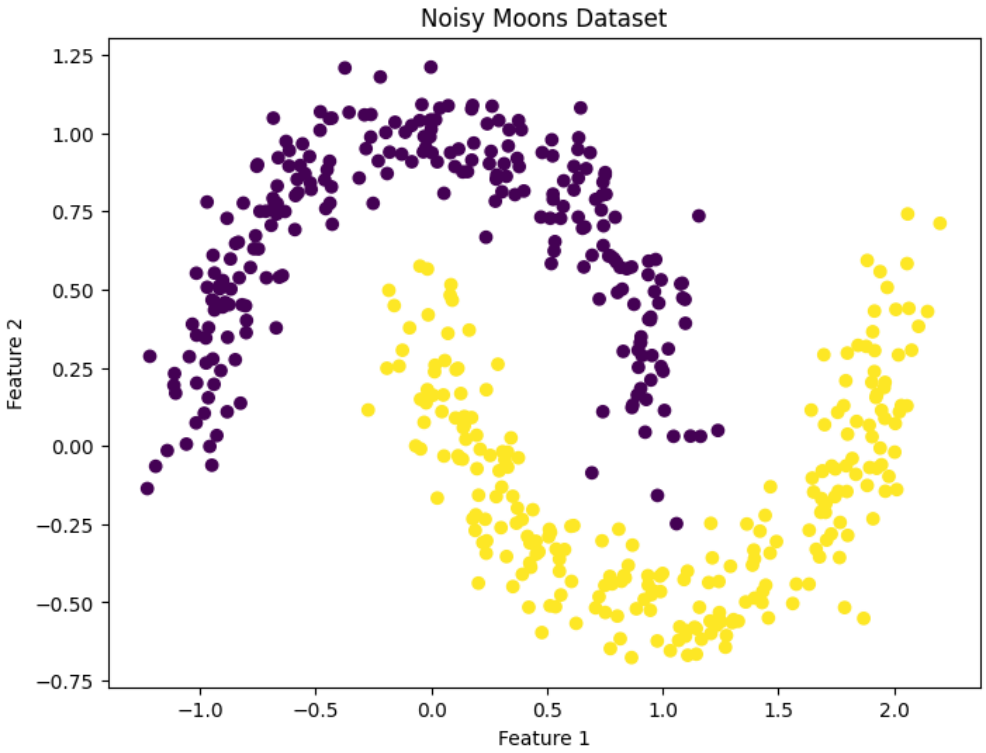

Figure 8: Scatter diagram of the Noisy Moons dataset.

Analysis of the Noisy Moons Dataset

Upon visual inspection, the dataset appears to have two clear, albeit noisy, moon-shaped clusters. This shape is characteristic of the "Noisy Moons" dataset and presents a non-trivial challenge for clustering algorithms due to the noise factor and the non-linear separation between the two classes.

Introduction

The 'Noisy Moons' dataset is a two-dimensional synthetic dataset ideal for demonstrating clustering algorithms. The dataset comprises two interleaving half-circles with added Gaussian noise, making it challenging for clustering algorithms to identify the underlying pattern.

Class Approximation with Normal Distribution

Each class in the dataset was approximated using a normal distribution. The parameters of these distributions, namely the mean and covariance matrix for each class, were calculated and used to draw the related contours.

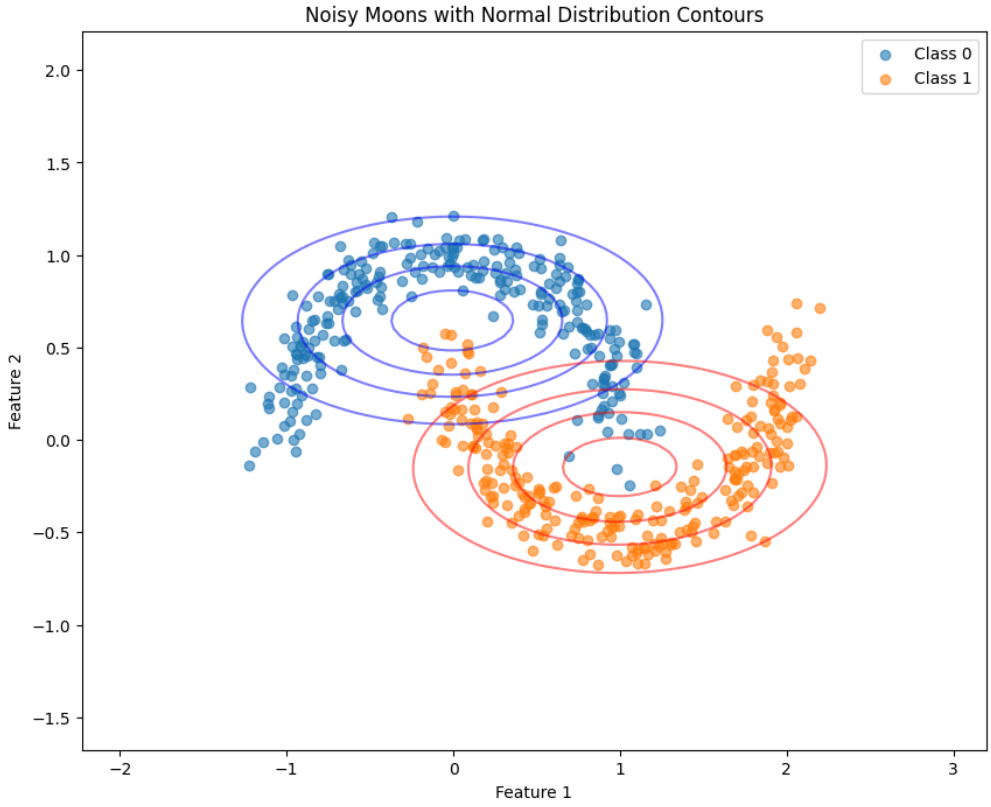

Figure 9: Normal distribution contours for each class.

Gaussian Mixture Model Implementation

A GMM was implemented manually, fitting models with the number of components ranging from 1 to 16. Contours for models with 3, 8, and 16 components were plotted to visualize the fit.

Approximation with Normal Distribution

Each class of the "Noisy Moons" dataset was approximated with a normal distribution. The parameters of these distributions, the mean vectors and covariance matrices, were obtained. The contours representing these distributions are overlaid on the scatter plot of the dataset.

Parameters for Class Approximations

The estimated parameters for the normal distributions are:
Class 0:

$$\text{Mean} = \begin{bmatrix} -0.00549138 \\ 0.64610592 \end{bmatrix}$$

$$\text{Covariance Matrix} = \begin{bmatrix} 0.52479732 & 0.00116374 \\ 0.00116374 & 0.1039425 \end{bmatrix}$$

Class 1:

$$\text{Mean} = \begin{bmatrix} 0.99931885 \\ -0.14663469 \end{bmatrix}$$

$$\text{Covariance Matrix} = \begin{bmatrix} 0.51216003 & 0.0033396 \\ 0.0033396 & 0.10933026 \end{bmatrix}$$

Implementation Analysis

The normal distribution contours provide a visual understanding of the data's underlying distribution. The parameters were calculated manually, and the contours were generated to represent the Gaussian distributions that best fit each class of the dataset. This visualization is crucial for recognizing the patterns and spread of the data, which informs the clustering process when applying Gaussian Mixture Models.

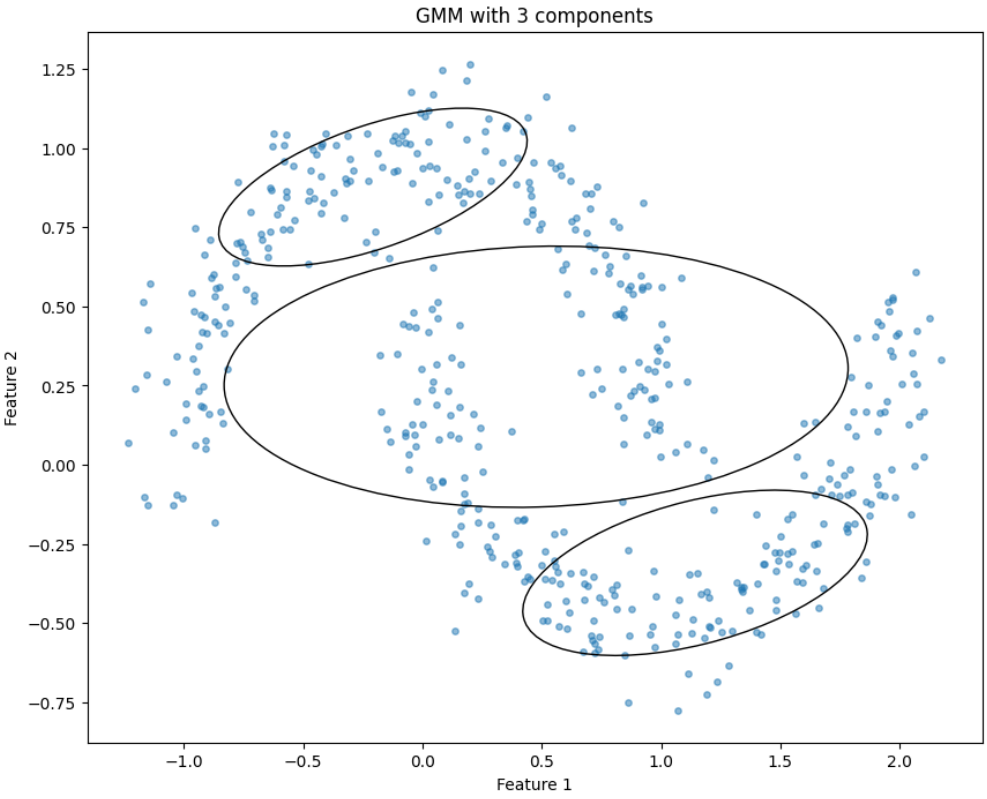

Figure 10: GMM contour with 3 components.

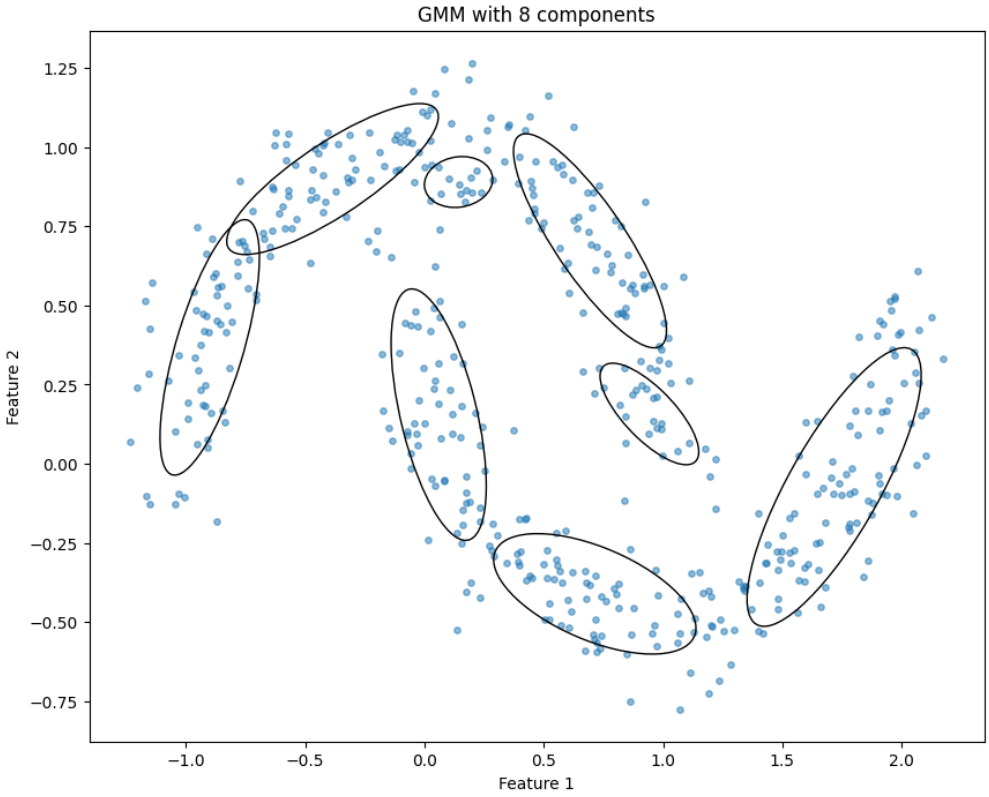

Figure 11: GMM contour with 8 components.

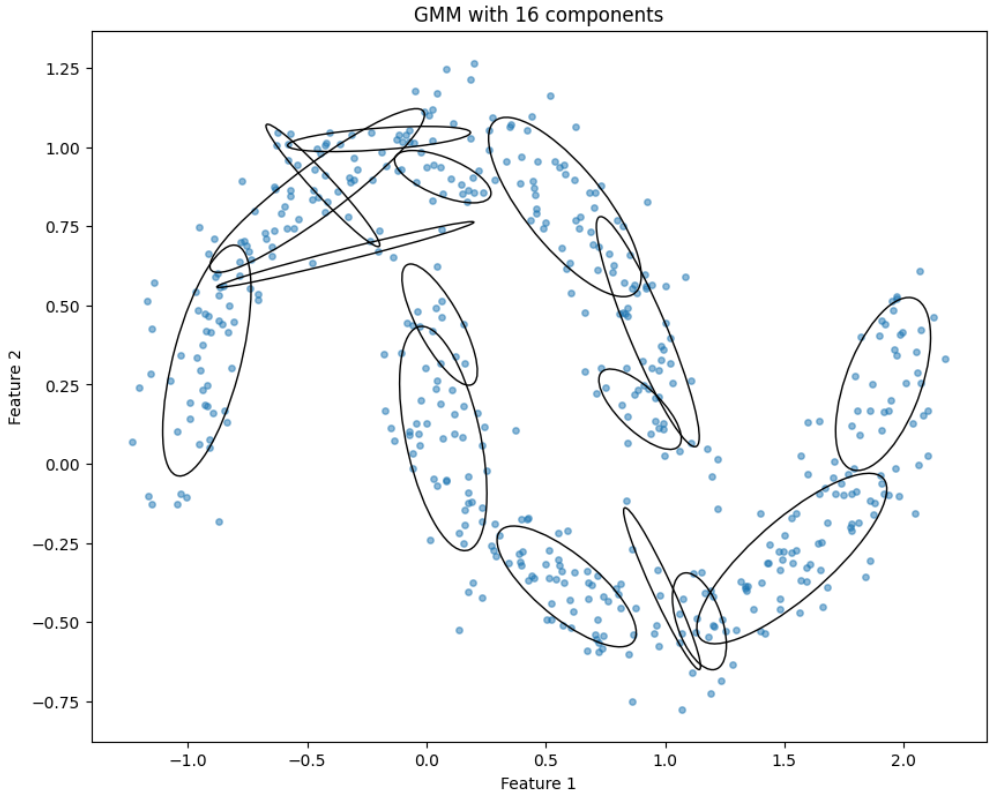

Figure 12: GMM contour with 16 components.

Optimization of Model Components

To determine the optimal number of components for the Gaussian Mixture Model, we evaluated models with different numbers of components using the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC). The AIC and BIC are both penalized-likelihood criteria and serve to balance model fit with model complexity.

AIC and BIC Scores

The AIC and BIC scores for models with 1 to 16 components were calculated. A lower score on these metrics generally indicates a model with a better fit. We found that the model with 10 components had the lowest AIC and BIC scores, suggesting it as the most suitable model for our dataset. The AIC and BIC scores for the model with 10 components are as follows:

- AIC: 1191.0279020237817
- BIC: 1349.3771560169746

The plot below (Figure 13) illustrates the AIC and BIC scores across different model complexities.

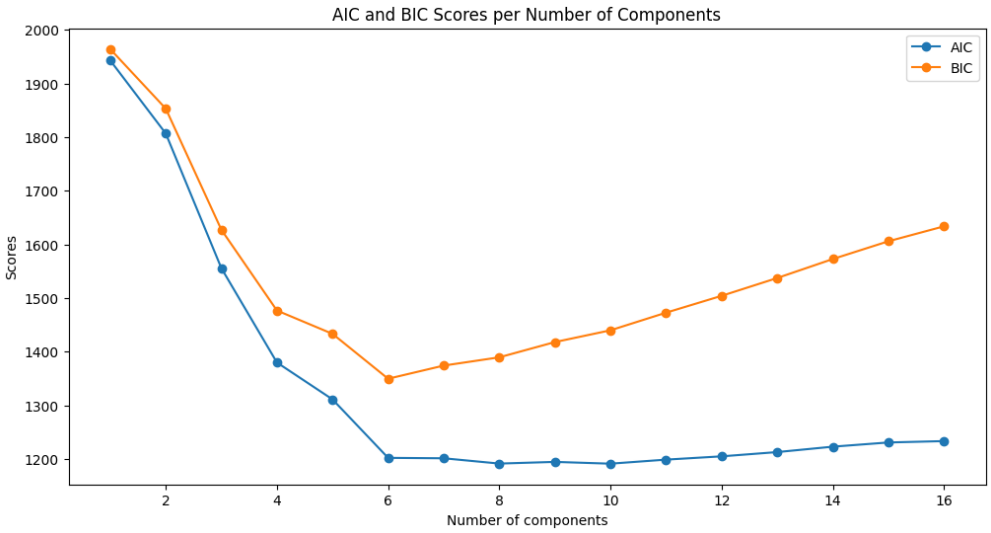

Figure 13: AIC and BIC scores per number of components in the Gaussian Mixture Model.

Analysis

The trend in the AIC and BIC values suggests that increasing the number of components initially provides a better fit, as indicated by the sharp decrease in both AIC and BIC values. However, beyond a certain point, the additional components do not contribute significantly to improving the model, which is reflected in the plateau and subsequent increase in the BIC values. This behavior is consistent with the principle of parsimony, as overly complex models are penalized under these criteria. Based on the AIC and BIC metrics, we select a Gaussian Mixture Model with 10 components as the optimal model for the Noisy Moons dataset.