*University of Tehran*

*School of Electrical and Computer Engineering*

# Machine Learning

Homework 2: Regression and Non-Parametric Estimation

*Author:*
Alireza Javid

*Student Number:*
810198375

# Contents

# 1 Problem 1: Linear Regression

In this problem, we set $x$ values for the amount of study and $y$ values for the grades.

(a)

$$y_i = \beta_0 + \varepsilon_i \quad \rightarrow \quad J(\beta) = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} (y_i - \beta_0)^2$$

$$\frac{\partial J(\beta)}{\partial \beta} = \sum_{i=1}^{n} (\beta_0 - y_i) = 0 \quad \rightarrow \quad \beta_0 = \frac{1}{n} \sum_{i=1}^{n} y_i$$

$$\beta_0 = \frac{1}{10}\big(34+47+66+52+80+35+66+48+87+81\big) = 59.6$$

(b)

$$y_i = \beta_1 x_i + \varepsilon_i \quad \rightarrow \quad J(\beta) = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} (y_i - \beta_1 x_i)^2$$

$$\frac{\partial J(\beta)}{\partial \beta} = \sum_{i=1}^{n} x_i(\beta_1 x_i - y_i) = 0 \quad \rightarrow \quad \beta_1 = \frac{\sum_{i=1}^{n} y_i x_i}{\sum_{i=1}^{n} x_i^2}$$

$$= \frac{34(5)+47(11)+66(18)+52(15)+80(21)+35(6)+66(17)+48(10)+87(24)+81(19)}{5^2+11^2+18^2+15^2+21^2+6^2+17^2+10^2+24^2+19^2}$$

$$= \frac{9774}{2498} = 3.913$$

(c) The first equation is a regression line that describes the straight correlation between the average of the fitted value and the explanatory variable $X$. The second equation is a linear model that represents the relationship between the observed pairs $(X,Y)$. Unlike the first equation, not all pairs will be directly on a line due to the presence of an error term ( $\varepsilon$ ). The values $b_0$ and $b_1$ in the first equation are determined from the data, while the parameters $\beta_0$ and $\beta_1$ in the second equation are not known.

(d) We define the sum of squared error(SSE) as below:

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)$$

Where $\hat{y}_i = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p$. As we know, the value of $\beta_i$ is unknown and we had to estimate all of $\beta_0, \beta_1, ..., \beta_p$, which are $p+1$ parameters. So due to the concept of degrees of freedom, now an unbiased estimator is

$$\sigma^2 = \frac{SSE}{n-p-1}$$

And when we have $p = 1$ (above example), the variance estimator is

$$\sigma^2 = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)}{n-2}$$

(e) Not necessarily. The new observation is a *random variable* and if we assume the distribution of linear regression error for this problem is a Normal random variable, the estimated value is just the mean value of this random variable and the actual value would be different.

# 2 Problem 2: Regularization

Regularization refers to techniques that are used to calibrate machine learning models in order to minimize the adjusted loss function and prevent overfitting or underfitting.

(a)

- L1 regularization is also referred to as the L1 norm or Lasso, we shrink the parameters to zero. When input features have weights closer to zero that leads to a sparse L1 norm. In a sparse solution, the majority of the input features have zero weights and very few features have non-zero weights.

  In L1 regularization we penalize the absolute value of the weights. In the following equation consider $\hat{y}_i = \omega_0 + \omega_1 x_1 + \ldots + \omega_p x_p$ and $\omega = \begin{bmatrix} \omega_0 \\ \omega_1 \\ \vdots \\ \omega_p \end{bmatrix}$

$$L(x,y) \equiv \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{i=1}^{n} ||\omega||_1$$

- In L2 regularization, the regularization term is the sum of squares of all feature weights as shown in the equation. L2 regularization forces the weights to be small but does not make them zero and does a non-sparse solution. L2 is not robust to outliers as square terms blow up the error differences of the outliers and the regularization term tries to fix it by penalizing the weights.

$$L(x,y) \equiv \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{i=1}^{n} ||\omega||_2^2$$

- In L1 regularization, the sum of absolute values of the weights is penalized, while in L2 regularization, the sum of squares of the weights is penalized. The solution obtained using L1 regularization is often sparse, meaning that many of the weights are set to zero, while the solution obtained using L2 regularization is non-sparse, meaning that all of the weights are used. L1 generates models that are simple and interpretable but cannot learn complex patterns, whereas L2 regularization is able to learn complex data patterns. Moreover, L1 is robust to outliers and has more than one solution but L2 is not robust to outliers and has one solution.

This article has illustrated L1 and L2 regularization with more details.

(b)

$$\hat{\beta} = \arg\min_{\beta} (Y - X\beta)^2 + \lambda ||\beta||_2^2$$

For minimizing the objective function:

$$\frac{\partial}{\partial \beta}\left((Y - X\beta)^T (Y - X\beta) + \lambda \beta^T \beta\right) = \frac{\partial}{\partial \beta}\left(Y^T Y - 2\beta^T X^T Y + \beta^T X^T X \beta + \lambda \beta^T I \beta\right)$$

$$= 2X^T X\beta - 2X^T Y + 2\lambda I\beta = 0$$

$$\hat{\beta} = X^T Y \left(X^T X + \lambda I\right)^{-1}$$

As we can see, the ( $\lambda I$ ) term is added to the ordinary linear regression solution.

# 3 Problem 3: Logistic Regression

$$P(Y = y_k \mid X) \sim \exp\left(w_{k_0} + \sum_{i=1}^{d} w_{k_i} X_i\right) \qquad \text{for } k = 1, \ldots, K-1$$

(a) First we assume $X = \begin{bmatrix} 1 \\ X_1 \\ \cdots \\ X_d \end{bmatrix}$ and $W_k = \begin{bmatrix} w_{k_0} \\ w_{k_1} \\ \cdots \\ w_{k_d} \end{bmatrix}$. Now we can rewrite the above relation as:

$$P(Y = y_k \mid X) \sim \exp\left(W_k^T X\right) \qquad \text{for } k = 1, \ldots, K-1$$

As we know the total probability is equal to 1 and by inspiring from the binary case we have:

$$log\Big(\frac{P(Y = y_k \mid X)}{1 - P(Y = y_k \mid X)}\Big) = \sum_{k=1}^{K-1} W_k^T X \quad \rightarrow \quad P(Y = y_k \mid X) = \frac{e^{W_k^T X}}{1 + \sum_{k=1}^{K-1} e^{W_k^T X}}$$

$$P(Y = y_k \mid X) = \frac{\exp(w_{k_0} + \sum_{i=1}^{d} w_{k_i} X_i)}{1 + \sum_{k=1}^{K-1} \exp\left(w_{k_0} + \sum_{i=1}^{d} w_{k_i} X_i\right)} \qquad \text{for } k = 1, \ldots, K-1$$

(b) As we have seen in the binary case we look at the $k$ value that maximizes the discriminant function. In other words, the classification rule selects the label with the highest probability.

$$k^* = \arg\max_{i=1}^{K-1} P(Y = y_i | X)$$

# 4 Problem 4: Parzen Density Estimation

$$P_n(x) = \frac{1}{nh_n}\sum_{i=1}^{n}\varphi\left(\frac{x-x_i}{h}\right) \quad , \; P(x) \sim N(\mu,\sigma^2) \; , \; \phi(x) \sim N(0,1)$$

(a)

$$\tilde{P}_n(x) = E[P(x)] = \frac{1}{n}E\Big(\frac{1}{h_n}\sum_{i=1}^{n}\varphi\left(\frac{x-x_i}{h}\right)\Big) = \frac{1}{n}\sum_{i=1}^{n}E\Big(\frac{1}{h_n}\varphi\left(\frac{x-x_i}{h}\right)\Big)$$

$$= E\Big(\frac{1}{h_n}\varphi\left(\frac{x-x_i}{h}\right)\Big) = \int \frac{1}{h_n}\varphi\left(\frac{x-x_i}{h}\right)P(x_i)dx_i$$

$$= \int \frac{1}{2\pi h_n\sigma}\exp\left(\frac{-1}{2}\Big(\frac{(x_i-x)^2}{2h_n^2}+\frac{(x_i-\mu)^2}{2\sigma^2}\Big)\right)dx_i$$

$$= \frac{1}{2\pi h_n\sigma}\exp\left(\frac{x^2}{2h_n^2}+\frac{\mu^2}{2\sigma^2}\right)\int \exp\left(\frac{-1}{2}\Big(x_i^2(\frac{1}{h_n^2}+\frac{1}{\sigma^2})-2x_i(\frac{x}{h_n^2}+\frac{\mu}{\sigma^2})\Big)\right)dx_i$$

$$= \frac{1}{2\pi h_n\sigma}\exp\left(\frac{-x^2}{2h_n^2}+\frac{-\mu^2}{2\sigma^2}+\frac{\omega^2}{2s^2}\right)\int \exp\left(\frac{-1}{2}(\frac{x_i-\omega}{s})^2\right)dx_i$$

In which $\begin{cases} s^2 = \frac{1}{\frac{1}{h_n^2}+\frac{1}{\sigma^2}} \\ \omega = s^2(\frac{x}{h_n^2}+\frac{\mu}{\sigma^2}) \end{cases}$

$$\tilde{P}_n(x) = \frac{s}{\sqrt{2\pi}h_n}\exp\left(\frac{-1}{2}\Big(\frac{x^2}{h_n^2}+\frac{\mu^2}{\sigma^2}-\frac{\omega^2}{s^2}\Big)\right)$$

$$= \frac{h_n\sigma}{\sqrt{2\pi(h_n^2+\sigma^2)}h_n}\exp\left(\frac{-1}{2}\Big(\frac{x^2}{h_n^2}+\frac{\mu^2}{\sigma^2}-(\frac{x^2\sigma^2}{h_n^2(\sigma^2+h_n^2)}+\frac{\mu^2h_n^2}{\sigma^2(\sigma^2+h_n^2)}+\frac{2x\mu}{(\sigma^2+h_n^2)})\Big)\right)$$

$$= \frac{h_n\sigma}{\sqrt{2\pi(h_n^2+\sigma^2)}h_n}\exp\left(\frac{-1}{2}\Big(\frac{x^2}{h_n^2}(1-\frac{\sigma^2}{\sigma^2+h_n^2})+\frac{\mu^2}{\sigma^2}(1-\frac{h_n^2}{\sigma^2+h_n^2})-\frac{2x\mu}{\sigma^2+h_n^2}\Big)\right)$$

$$\tilde{P}_n(x) = \frac{1}{\sqrt{2\pi(h_n^2+\sigma^2)}}\exp\left(\frac{-1}{2}\Big(\frac{(x-\mu)^2}{h_n^2+\sigma^2}\Big)\right)$$

As we can see $\tilde{P}_n(x) \sim N(\mu,\, h_n^2+\sigma^2)$

(b)

$$P(x)-\tilde{P}_n(x) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)-\frac{1}{\sqrt{2\pi(h_n^2+\sigma^2)}}\exp\left(\frac{-1}{2}\Big(\frac{(x-\mu)^2}{h_n^2+\sigma^2}\Big)\right)$$

$$= P(x)\left(1-\frac{1}{\sqrt{1+(\frac{h_n}{\sigma})^2}}\exp\left(\frac{-(x-\mu)^2}{2}\Big(\frac{1}{\sigma^2+h_n^2}-\frac{1}{\sigma^2}\Big)\right)\right)$$

$$= P(x)\left(1-\frac{1}{\sqrt{1+(\frac{h_n}{\sigma})^2}}\exp\left(\frac{(x-\mu)^2}{2}\frac{h_n^2}{\sigma^2(\sigma^2+h_n^2)}\right)\right)$$

The problem indicates, $h_n$ is relatively small, so we can use exponential and fraction approximation as below:

$$\approx P(x)\left(1-\left(1-\frac{1}{2}(1+(\frac{h_n}{\sigma})^2)\right)\left(1+\frac{(x-\mu)^2}{2}\frac{h_n^2}{\sigma^2(\sigma^2+h_n^2)}\right)\right)$$

We can omit the terms, having $h_n^4$ and rewrite the above equation

$$\approx P(x)\left(\frac{h_n}{\sigma}\right)^2\left(1-(\frac{x-\mu}{\sigma})^2\right)$$

# 5 Problem 5: K-NN Classification

To prove the resulting space is a metric space, we should prove the three main characteristics of metric space.

$$D(\mathbf{a},\mathbf{b}) = \sqrt{\sum_{k=1}^{d} x_k^2\,(a_k - b_k)^2} \quad x_k \neq 0$$

- We know that $x_k^2 > 0$ and $(a_k - b_k)^2 \geq 0$. Due to that, we can simply say $D(\mathbf{a},\mathbf{b}) \geq 0$ and $\geq 0$ and $D(\mathbf{a},\mathbf{b}) = 0$ if and only if $a = b$.
- It's obvious $D(\mathbf{a},\mathbf{b}) = D(\mathbf{b},\mathbf{a})$. Due to the fact that $(a-b)^2 = (b-a)^2$.
- At last, we should prove triangle inequality. $D(\mathbf{a},\mathbf{b}) + D(\mathbf{b},\mathbf{c}) \geq D(\mathbf{a},\mathbf{c})$.

  We can say $D(\mathbf{a},\mathbf{b}) = ||A(\mathbf{a}-\mathbf{b})||_2$, where $A = diag(x_1,...,x_d)$

  $$||A(\mathbf{a}-\mathbf{b})||_2 + ||A(\mathbf{b}-\mathbf{c})||_2 \geq ||A(\mathbf{a}-\mathbf{c})||_2$$

  We can define $X = A(\mathbf{a}-\mathbf{b})$ and $Y = A(\mathbf{b}-\mathbf{c})$.

  $$||X||_2 + ||Y||_2 + \geq ||X+Y||_2$$

  $$(||X||_2 + ||Y||_2)^2 = ||X||_2^2 + ||Y||_2^2 + 2||X||_2||Y||_2$$

  $$||X+Y||_2^2 = ||X||_2^2 + ||Y||_2^2 + 2\sum_{k=1}^{d} X_k Y_k$$

  From Cauchy-Schwarz inequality we know that $||X||_2||Y||_2 \geq \left|\sum_{k=1}^{d} X_k Y_k\right|$

  $$||X||_2^2 + ||Y||_2^2 + 2||X||_2||Y||_2 \geq ||X||_2^2 + ||Y||_2^2 + 2\sum_{k=1}^{d} X_k Y_k$$

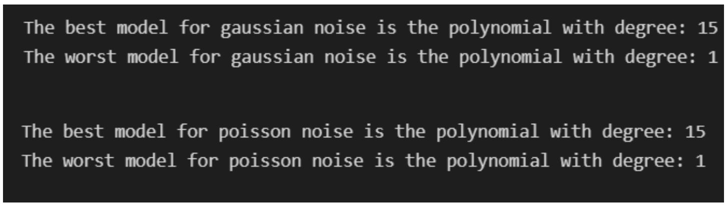
  $$||X||_2 + ||Y||_2 \geq ||X+Y||_2$$

In the nearest-neighbor method, if we use metric spaces, we can guarantee that there are the best approximations in this space. This means that we can find the closest point in a metric space to a given point by using the nearest-neighbor method. In other words, there is always a $\mathbf{p}$, which has the least distance from the input pattern $\mathbf{x}^*$ that implies:

$$p(\omega_i \mid \mathbf{p}) \approx p(\omega_i \mid \mathbf{x}^*)$$

# 6 Problem 6: Overfitting & Underfitting

(a) We consider MSE to select the best and the worst model. The results show that, When we use higher-degree functions, our mean square error becomes smaller. That's basically because *overfitting* occurs.



```
The best model for gaussian noise is the polynomial with degree: 15
The worst model for gaussian noise is the polynomial with degree: 1


The best model for poisson noise is the polynomial with degree: 15
The worst model for poisson noise is the polynomial with degree: 1
```

Figure 1: the best and the worst model based on MSE

(b) As below, we plot the fitted polynomial and then we calculate the MSE for each one.

- **Gaussian Noise:**

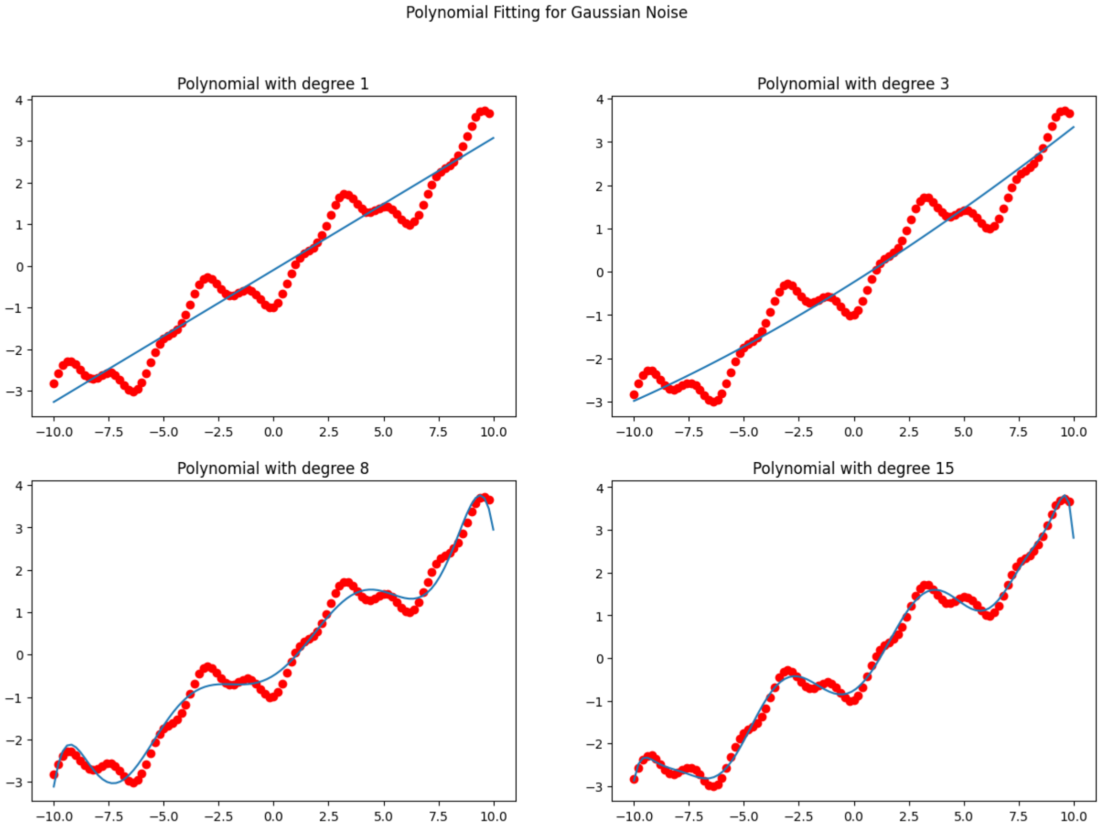
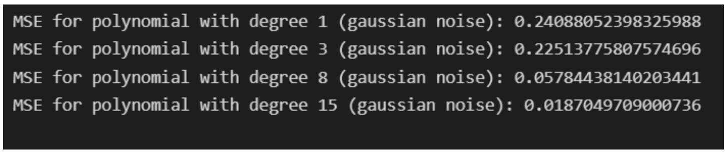


Figure 2: fitted polynomial for Gaussian noise



```
MSE for polynomial with degree 1 (gaussian noise): 0.24088052398325988
MSE for polynomial with degree 3 (gaussian noise): 0.22513775807574696
MSE for polynomial with degree 8 (gaussian noise): 0.05784438140203441
MSE for polynomial with degree 15 (gaussian noise): 0.0187049709000736
```

Figure 3: polynomial MSE values for Gaussian noise

- **Poisson Noise:**

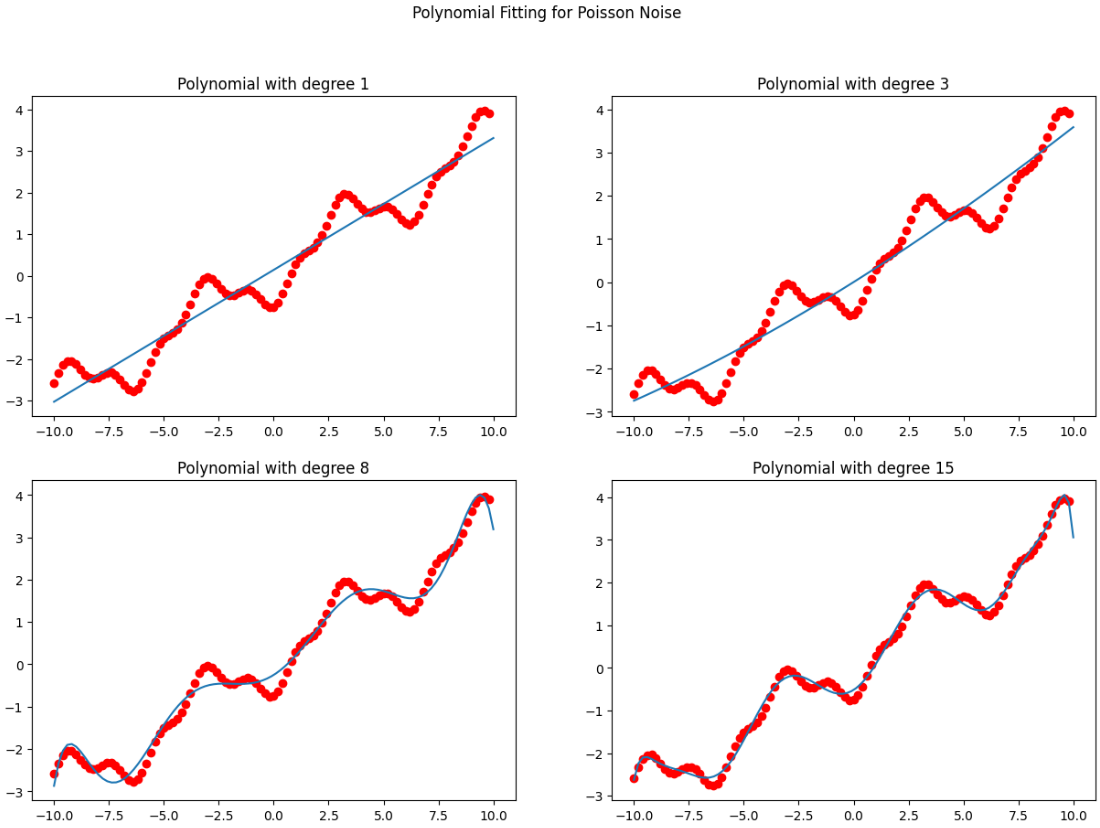
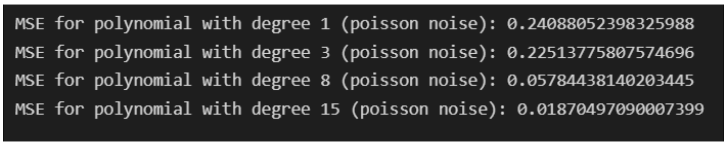


Figure 4: fitted polynomial for poisson noise



```
MSE for polynomial with degree 1 (poisson noise): 0.24088052398325988
MSE for polynomial with degree 3 (poisson noise): 0.22513775807574696
MSE for polynomial with degree 8 (poisson noise): 0.05784438140203445
MSE for polynomial with degree 15 (poisson noise): 0.018704970900007399
```

Figure 5: polynomial MSE values for poisson noise

(c) To calculate the bias and variance errors for the fitted polynomial models, we need to perform a Monte Carlo simulation. We consider 100 simulation numbers and report the mean value. Here are the resulting plots.

$$\sigma^2 = E[(\hat{f}(x) - E[\hat{f}(x)])^2] \quad , \quad baias^2 = E[(\hat{f}(x) - f(x))^2]$$

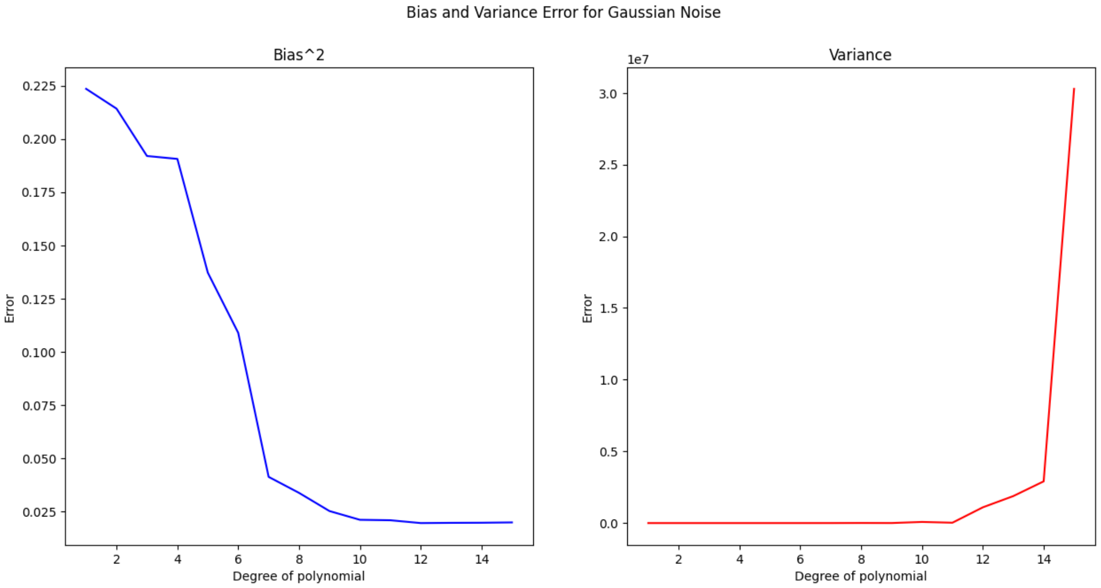


Figure 6: fitted polynomial for Gaussian noise

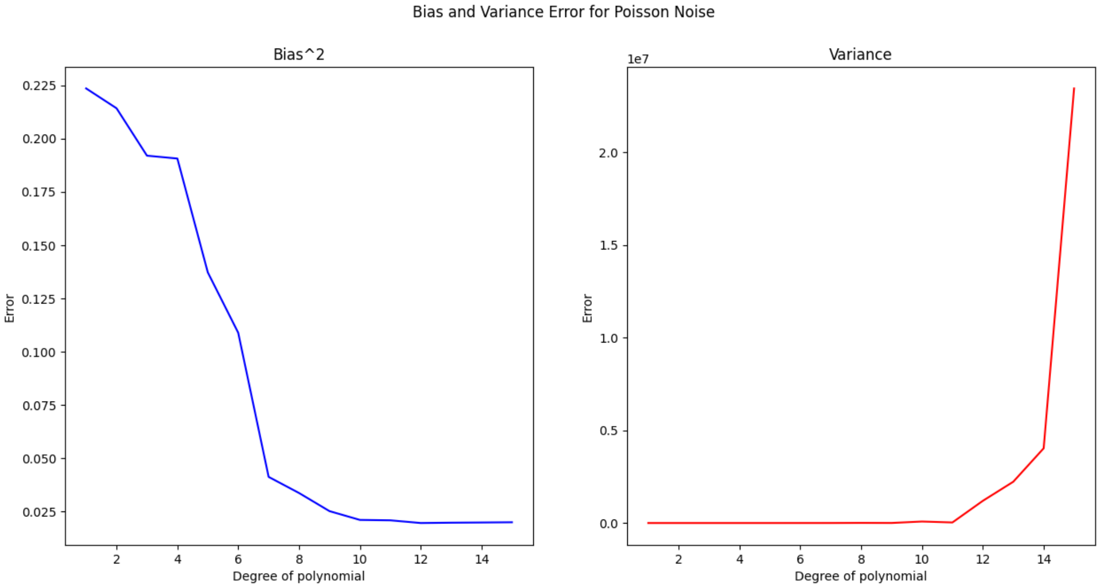


Figure 7: fitted polynomial for Poisson noise

In both Figures 6 and 7, we can observe that the polynomial with higher degrees tends to fit the training data better, resulting in a smaller bias error. However, as the degree of the polynomial increases, the model becomes more complex and the variance error increases. This is because the model tends to overfit the training data and cannot generalize well to new data, leading to a high variance error. Therefore, we can see the trade-off between bias and variance in this problem. In order to achieve the best performance, we need to find a balance between bias and variance errors, which can be achieved through techniques such as regularization or model selection.

# 7 Problem 7: Logistic Regression with L2 Regularization

(a) Below, we can see the distribution of data points in both states:

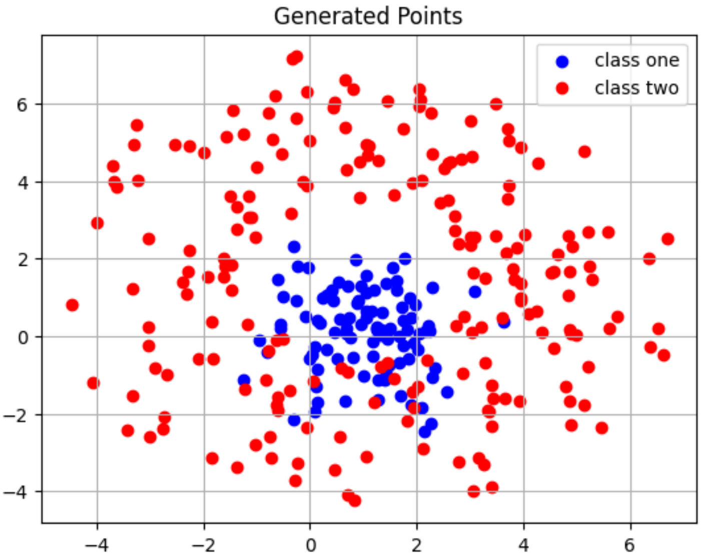


Figure 8: data points of state 1

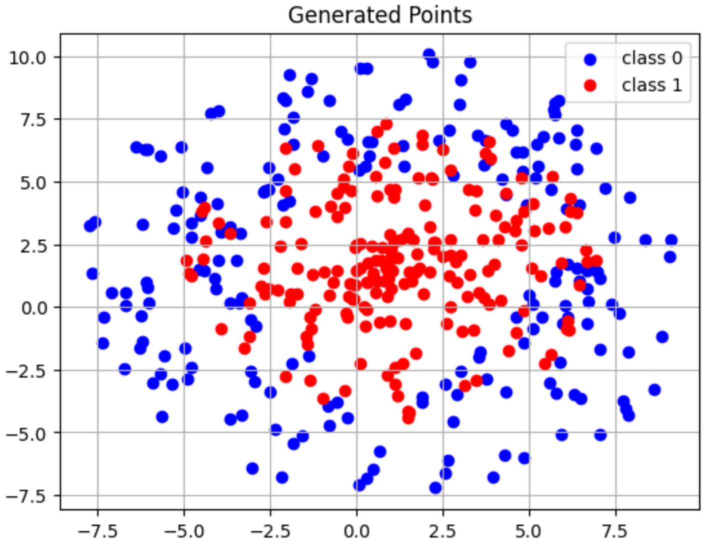


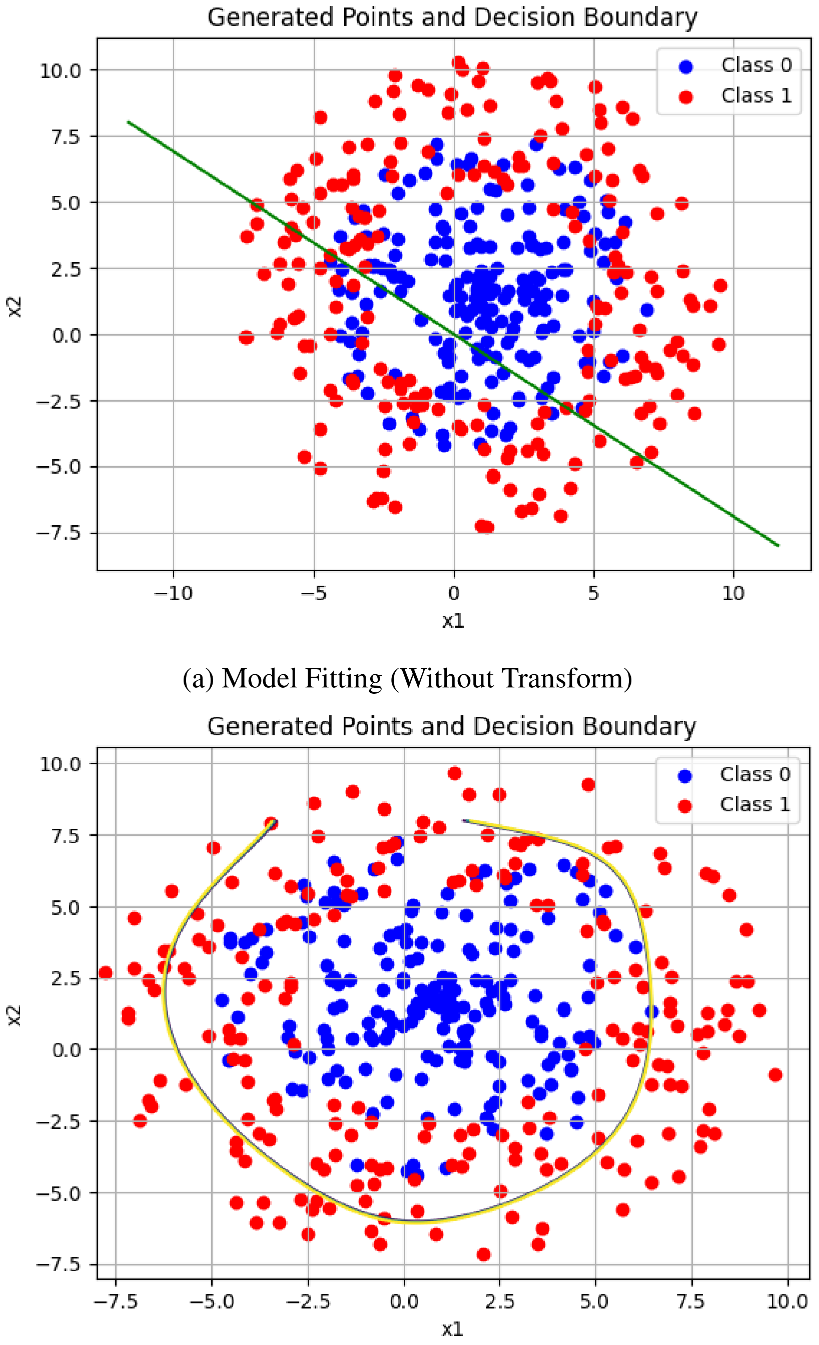
Figure 9: data points of state 2

(b) We design a class for *logistic regression* that can first transform data to a higher space and then make the classification. We use the below cost function and update rule for gradient descent:

$$J(\theta) = \frac{-1}{m} \sum_{i=1}^{m} \left[ y^{(i)} \log\left( h_\theta\left( x^{(i)} \right) \right) + \left( 1 - y^{(i)} \right) \log\left( 1 - h_\theta\left( x^{(i)} \right) \right) \right] + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left[ \left( h_\theta\left( x^{(i)} \right) - y^{(i)} \right) x_j^{(i)} + \lambda \theta_j \right]$$

You can see the code details in the attached file.

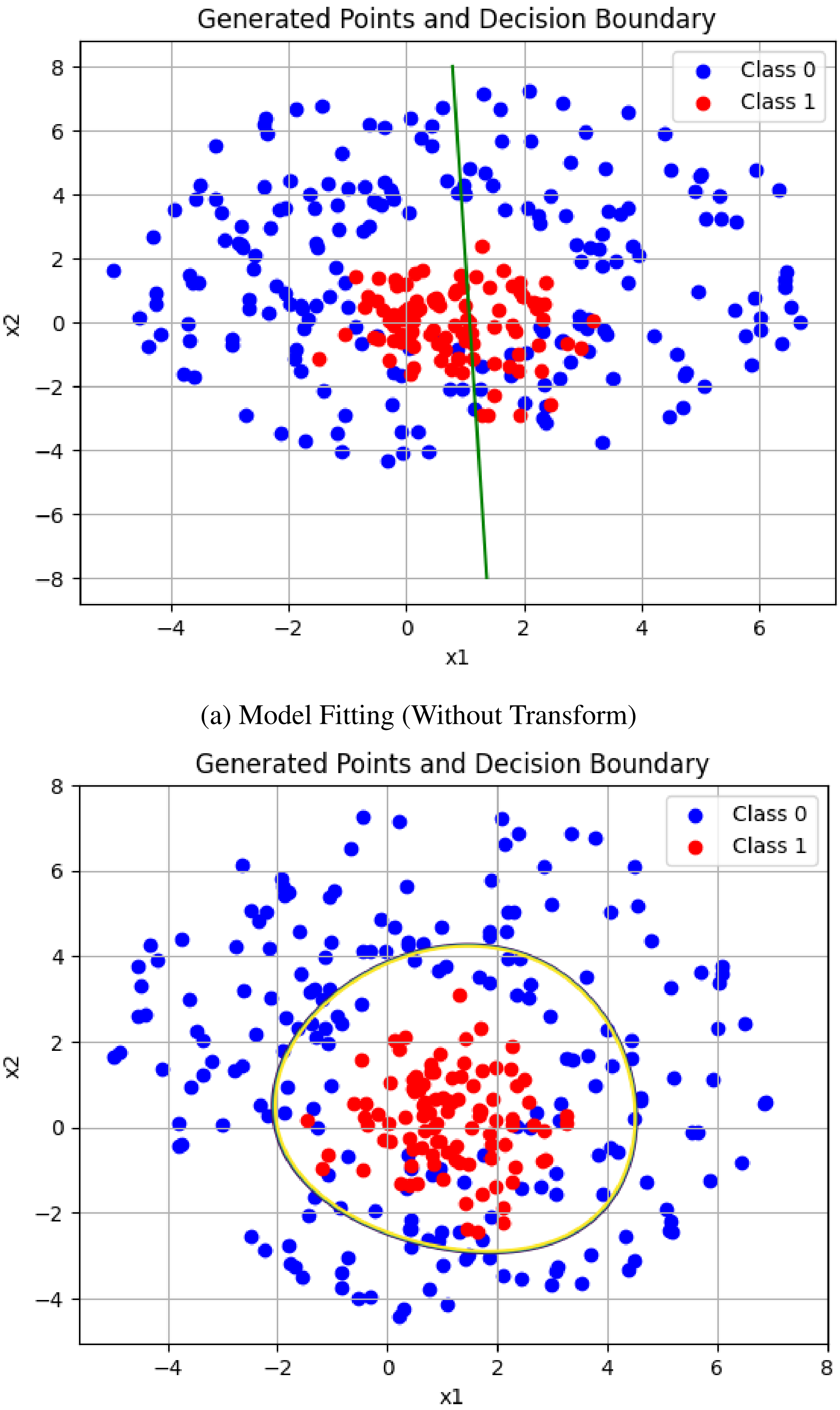


(a) Model Fitting (Without Transform)

(b) Model Fitting (Transform Data to High-Dimensional)

Figure 10: Model Fitting for State 1

The calculated accuracy without transformation is 0.56 and with transformation is 0.835. This shows in higher dimensions two classes are more linear separable and complex models have smaller bias (simplicity) errors.



(a) Model Fitting (Without Transform)

(b) Model Fitting (Transform Data to High-Dimensional)

Figure 11: Model Fitting for State 2

The calculated accuracy without transformation is 0.587 and with transformation is 0.92. This result is the same as the previous state.

(c) As we have said earlier, in higher dimensions, two classes are more linearly separable due to the increased number of features that can help distinguish between them. However, this can also lead to a higher variance (complexity) error if the model is too complex and overfits the training data. Therefore, finding the right balance between simplicity and complexity is crucial in achieving a low bias and variance error. In general, more complex models tend to have smaller bias errors but larger variance errors, while simpler models have larger bias errors but smaller variance errors. Finding the optimal trade-off between bias and variance is a key challenge in machine learning.

# 8 Problem 8: Parzen Density Estimation

(a) Parzen method uses the below formula to estimate the probability density function:

$$p_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{V_n}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)$$

In this problem, we have $\varphi(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}$ and $V_n = h_n$ and we use ted talks dataset.

First, we plot the density estimation with $h_n = 10$ and we have:

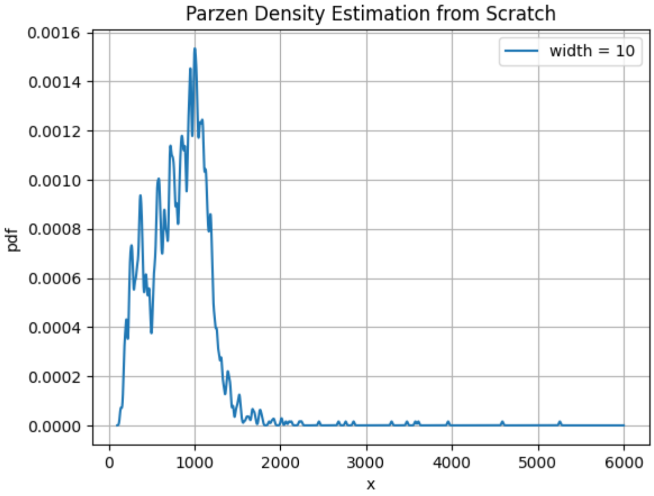


Figure 12: Parzen Density Estimation with $h_n = 10$

You can see the code details in the attached file.

(b) Now we try to find the relation between the final estimation and $h_n$. We consider three different values 20, 50, and 100.

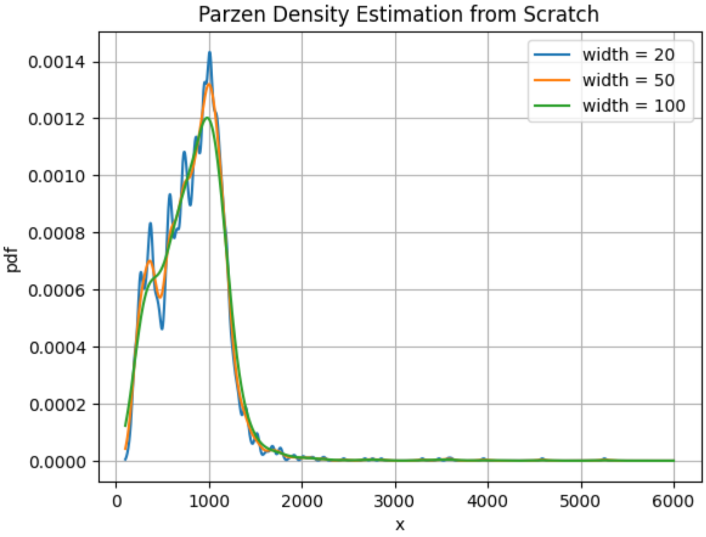


Figure 13: Parzen Density Estimation with three different $h_n$

As we can see, increasing the Parzen window results in a smoother density function with a decrease in the number of peaks and valleys in the distribution. This is because the Parzen window is a smoothing kernel used to estimate the density of a sample. By increasing the width of the window, more data points are included in each kernel density estimate, resulting in a smoother estimate with fewer abrupt changes in the density function. However, increasing the window size too much can result in over-smoothing and a loss of important features in the data. Therefore, choosing an appropriate window size is an important consideration when using the Parzen window method.

(c) If we use KDE to plot the density of the "duration" column, we would see the KDE function use higher width and it's more like our estimation with $h_n = 100$

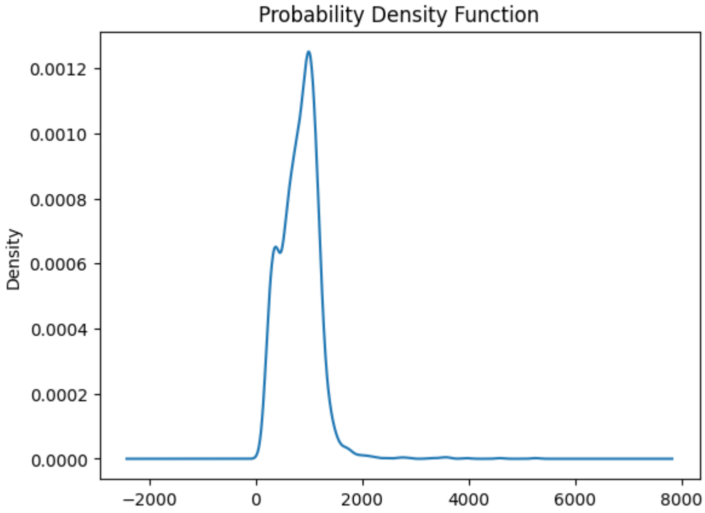


Figure 14: "duration" column distribution

In the figure provided below, we can observe the convergence of the estimated PDF to its final value as we increase the sample size. This indicates that the estimated PDF becomes more accurate with increasing sample size, which is a desirable property of any statistical estimator.

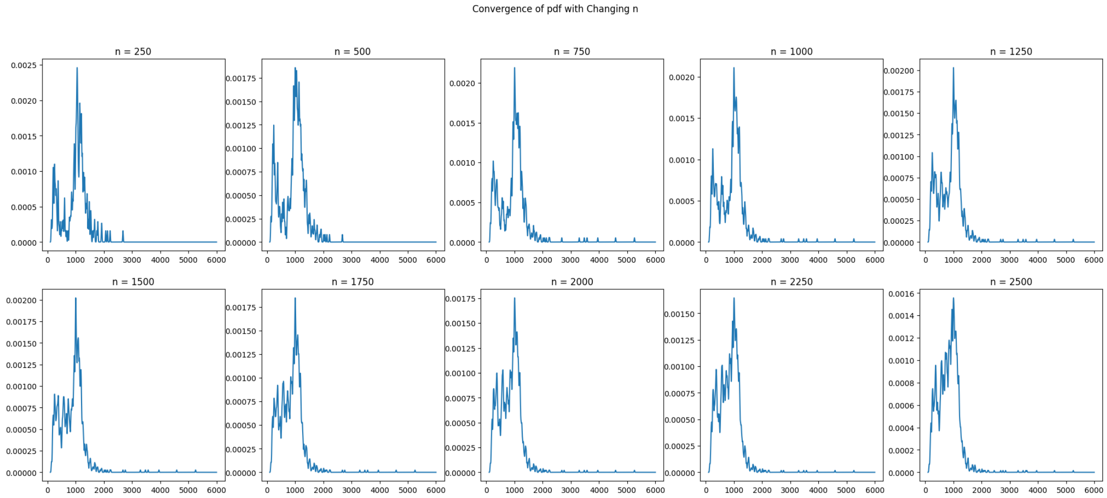


Figure 15: convergence of Parzen density estimation with changing $n$

(d) If we use Sklearn for Parzen density estimation which we calculate in part (a) we have the below picture which is quite similar.

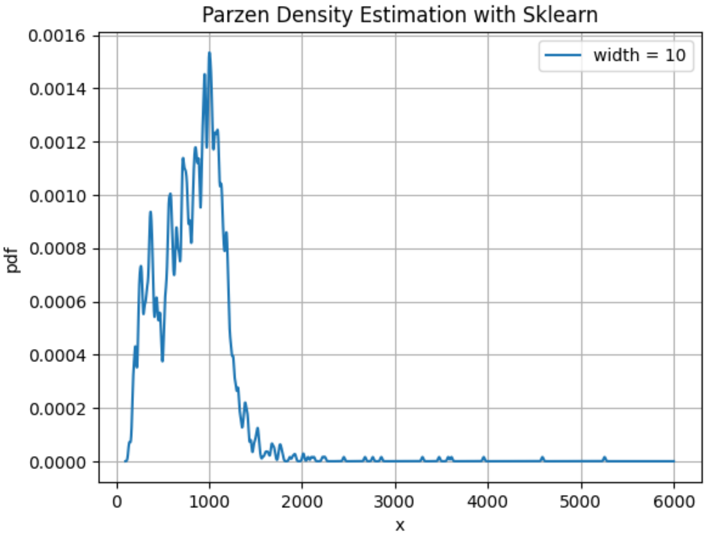


Figure 16: Parzen Density Estimation with $h_n = 10$ using Sklearn

Using Sklearn, we can show the convergence of Parzen density estimation with growing sample points.

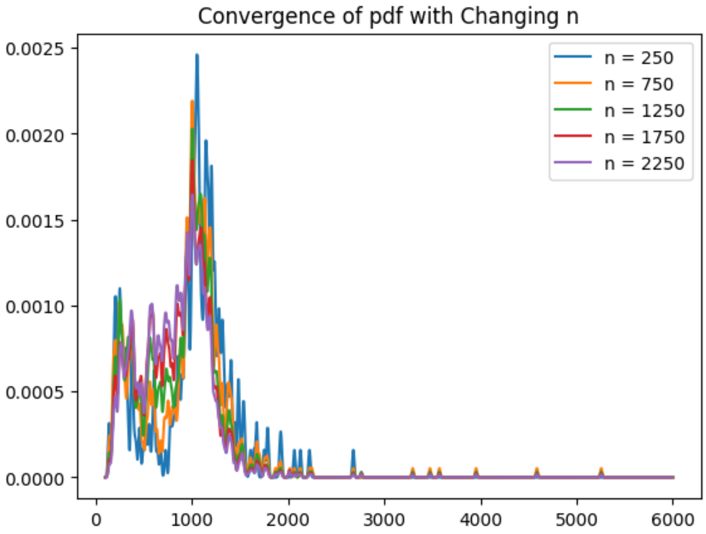


Figure 17: convergence of Parzen density estimation with changing *n* using Sklearn