DS-GA 1002 - Homework 2

Eric Niblock

September 15th, 2020

1. **(Half life) The half life of a radioactive substance is a way to quantify how rapidly the substance decays. Given a fixed quantity of the substance, the half time is the time that it takes for it to be reduced to half (i.e. half of the radioactive particles have decayed). It is not immediately apparent why the time should be the same for any quantity. Here we'll show that it is (probabilistically) if the particles decay following an exponential distribution.**

   (a) **Let $\tilde{t}$ be a random variable with a pdf of the form**

$$f_{\tilde{t}}(t) = \begin{cases} \lambda e^{-\lambda t}, & t \geq 0 \\ 0 & otherwise \end{cases} \tag{1}$$

where λ is a fixed constant. We define the half life $t_{1/2}$ as the number that satisfies $P(\tilde{t} > t_{1/2}) = \frac{1}{2}$. Compute $t_{1/2}$ in terms of λ. Then explain intuitively why this is a reasonable definition for the half life.

We can integrate the PDF is order to obtain $P(\tilde{t} > t_{1/2})$,

$$P(\tilde{t} > t_{1/2}) = \int_{t_{1/2}}^{\infty} \lambda e^{-\lambda t}\, dt = -\lambda \frac{1}{\lambda} e^{-\lambda t}\Big|_{t_{1/2}}^{\infty} = e^{-\lambda t_{1/2}} = \frac{1}{2} \tag{2}$$

$$t_{1/2} = -\frac{ln(1/2)}{\lambda} = \frac{ln(2)}{\lambda} \tag{3}$$

This is a reasonable definition for half life, because after this time, there is only a 50% chance of finding that the particle has decayed. A better way of looking at half life would be with a large number of particles, such as in a kilogram of radioactive material. After a period of one half life, we would expect roughly 50% of the material to have decayed. The definition of half life that we found says just that, because integrating from the value of the half life to infinity yields 50%. In other words, half of the material's "life" has passed, and half remains.

(b) Compute t such that $P(t_{1/2} < \tilde{t} < t) = \frac{1}{4}$, and express it in terms of only $t_{1/2}$. Explain why the result is consistent with the intuitive meaning of half life.

Again, we can compute $P(t_{1/2} < \tilde{t} < t)$ by integrating the PDF using the appropriate bounds,

$$P(t_{1/2} < \tilde{t} < t) = \int_{t_{1/2}}^{t'} \lambda e^{-\lambda t}\, dt = -e^{-\lambda t}\Big|_{t_{1/2}}^{t'} = -e^{-\lambda t} + e^{-\lambda t_{1/2}} = \frac{1}{4} \tag{4}$$

$$\frac{1}{4} = e^{ln(1/2)} - e^{-\lambda t} \implies e^{-\lambda t} = \frac{1}{4} \implies t = -\frac{ln(1/4)}{\lambda} = 2t_{1/2} \tag{5}$$

This result is expected given our intuition of half life. After a period of one half life we would expect 50% of radioactive material to remain. After another half life, we would expect 50% of this 50% of material to remain. In other words, we would expect 25% of the overall material to remain, which is the probability that we were originally given.

(c) Compute $P(\tilde{t} > kt_{1/2})$ for any integer k. Again, explain why the result is consistent with the intuitive meaning of half life.

Again, we can compute $P(\tilde{t} > kt_{1/2})$ by integrating the PDF using the appropriate bounds,

$$P(\tilde{t} > kt_{1/2}) = \int_{kt_{1/2}}^{\infty} \lambda e^{-\lambda t}\, dt = -e^{-\lambda t}\Big|_{kt_{1/2}}^{\infty} = e^{-\lambda k t_{1/2}} = e^{k ln(1/2)} = \left(\frac{1}{2}\right)^k \tag{6}$$

This result is expected given our notions regarding half life. After each half life, we would expect half of the remaining material to decay. After one half life, we expect half of the material to remain; two half lives, one fourth; three half lives, one eighth; etc.. This is what our result shows.

2. (Measurements) You have access to the readings of a device that indicates whether a radioactive particle has decayed. However you do not get a continuous reading, you get a reading every second.

(a) A reasonable model for the time the particle takes to decay is that it is a random variable with pdf

$$f_{\tilde{t}}(t) = \begin{cases} \lambda e^{-\lambda t}, & t \geq 0 \\ 0 & otherwise \end{cases} \tag{7}$$

where λ is a fixed constant. Taking into account that the measurement device rounds up the time and outputs an integer number of seconds (if the time is 0.1 it outputs 1, if it is 13.4 it outputs 14), compute the pmf of the reading from the device. What kind of random variable is this?

The random variable that we are concerned with is discrete, and we will note it as $\tilde{x}$; it is equal to the value of t rounded up to the nearest integer. Then, we have,

$$\begin{aligned} p_{\tilde{x}}(x) = P(\tilde{x} = x) = P(x-1 \leq \tilde{t} \leq x) = \int_{x-1}^{x} f_{\tilde{t}}(x)\, dt \\ = -e^{-\lambda t}\Big|_{x-1}^{x} = e^{-\lambda x}(e^{\lambda} - 1) \end{aligned} \tag{8}$$

So therefore, we have the pmf of $\tilde{x}$,

$$p_{\tilde{x}}(x) = \begin{cases} e^{-\lambda x}(e^{\lambda} - 1), & x \in N \\ 0 & otherwise \end{cases} \tag{9}$$

(b) What is the PDF of the error between your reading and the true time of decay?

We define $\tilde{e}$ to be the random variable associated with the error between $p_{\tilde{x}}(x)$ and $f_{\tilde{t}}(t)$. Then, we begin by calculating the cdf of $\tilde{e}$, $F_{\tilde{e}}(y)$, where $0 \leq y \leq 1$ so,

$$F_{\tilde{e}}(y) = P(\tilde{e} \leq y) = P(\cup_{i=1}^{\infty}\{i - y \leq \tilde{t} \leq i\}) \tag{10}$$

And since these events are all disjoint, we can write,

$$\begin{aligned} F_{\tilde{e}}(y) &= \sum_{i=1}^{\infty} P(i-y \leq \tilde{t} \leq i) = \sum_{i=1}^{\infty} \int_{i-y}^{i} f_{\tilde{t}}(t)\,dt = \sum_{i=1}^{\infty} \int_{i-y}^{i} \lambda e^{-\lambda t}\,dt \\ &= \sum_{i=1}^{\infty} -e^{-\lambda t}\Big|_{i-y}^{i} = (e^{\lambda y}-1)\sum_{i=1}^{\infty} e^{-\lambda i} = (e^{\lambda y}-1)\sum_{i=0}^{\infty}(e^{-\lambda i}) - 1 \end{aligned} \tag{11}$$

This is simply a geometric series, so we have,

$$F_{\tilde{e}}(y) = (e^{\lambda y}-1)\left(\frac{1}{1-e^{-\lambda}} - 1\right) \tag{12}$$

And we can obtain the pdf for $\tilde{e}$ by taking the derivative of the cdf, so,

$$f_{\tilde{e}}(y) = \frac{dF_{\tilde{e}}(y)}{dy} = \lambda e^{\lambda y}\left(\frac{1}{1-e^{-\lambda}} - 1\right) \tag{13}$$

Where again, $0 \leq y \leq 1$. Otherwise, the pdf evaluates to 0.

3. (Call center) A company that runs a call center hires you as a consultant. They are interested in the probability of receiving a certain number of calls in any 10 minute interval between 8 pm and 12 am during October in order to decide whether to hire more operators. Your task is to estimate these probabilities from data provided by the company. In particular, you have available two data sets:

- ***2-day data*: Number of calls in 10-minute intervals between 8 pm and 12 am from two days at the beginning of October.**
- ***September data*: Number of calls in 10-minute intervals between 8 pm and 12 am from the whole month of September.**

You decide to estimate the distribution of the calls in October from these two data sets by using a parametric and a non-parametric approach.

(a) Telephone calls are often modeled using Poisson distributions. What is the ML estimator of the parameter of a Poisson random variable given n independent observations?

We can denote the Poisson distribution, with $\tilde{x}$ as the Poisson random variable, as,

$$p_{\tilde{x}}(x;\lambda) = P(\tilde{x} = x;\lambda) = \frac{\lambda^x e^{-\lambda}}{x!} \tag{14}$$

Then, given $x_1, ..., x_n$ observations, we have a likelihood function which is equal to,

$$\mathcal{L}_{\{x_1,\ldots,x_n\}}(\theta) = P(\theta = x_1, ...x_n;\theta) \tag{15}$$

And assuming independence of the observations, we then have,

$$\mathcal{L}_{\{x_1,\ldots,x_n\}}(\theta) = \prod_{i=1}^{n} P(\theta = x_i;\theta) = \prod_{i=1}^{n} \frac{\theta^{x_i} e^{-\theta}}{x_i!} \tag{16}$$

$$\begin{aligned} ln(\mathcal{L}_{\{x_1,\ldots,x_n\}}(\theta)) &= ln\left(\prod_{i=1}^{n}\frac{\theta^{x_i}e^{-\theta}}{x_i!}\right) = \sum_{i=1}^{n} ln\left(\frac{\theta^{x_i}e^{-\theta}}{x_i!}\right) \\ &= \sum_{i=1}^{n}\left(x_i ln(\theta) - \theta - ln(x_i!)\right) \\ &= -n\theta + ln(\theta)\sum_{i=1}^{n} x_i - \sum_{i=1}^{n} ln(x_i!) \end{aligned} \tag{17}$$

Now, in order to find what value of θ maximizes this likelihood function, we take a derivative with respect to θ and equate it to zero.

$$\frac{\partial ln(\mathcal{L}_{\{x_1,\ldots,x_n\}}(\theta))}{\partial\theta} = 0 = -n + \frac{1}{\theta_{ML}}\sum_{i=1}^{n} x_i \tag{18}$$

This implies that,

$$\theta_{ML} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{19}$$

(b) Complete the code in *call_center_poisson.py*. This code estimates the Poisson parameter corresponding to the 2-day data and the September data. It compares the corresponding pmfs to the pmf of October and computes the estimation error (using the ‘1 norm of the difference between the distributions). Then it computes the empirical pmf of the 2-day data and the September data and again compares them to the pmf of October. What errors do you obtain? Submit your plots and the code.

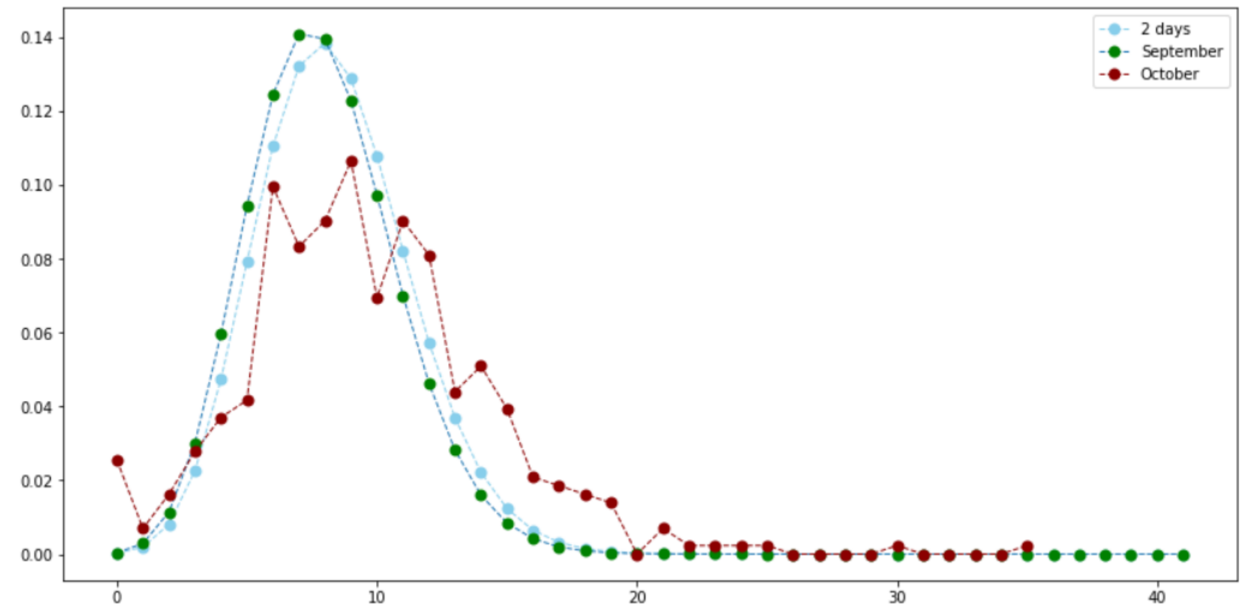
The following is the completed code from the *call_center_poisson.py* file. Only three definitions where manipulated:

```python
# Function to estimate the parameters of a Poisson r.v. from iid data
def poisson_ml_estimator(data):

    return(sum(data)/len(data))


# Function to estimate the empirical pmf from iid data
def empirical_pmf(data):

    ma = max(data)
    mi = min(data)
    pmf = []

    for d in range(int(mi), int(ma+1)):
        c = np.count_nonzero(data == d)
        pmf.append(c/len(data))

    return(np.array(pmf))


# Function to compute pmf of Poisson with parameter param at x
def poisson_pmf(param, x):
    return((param**x)*(np.exp(-1*param))/math.factorial(x))
```

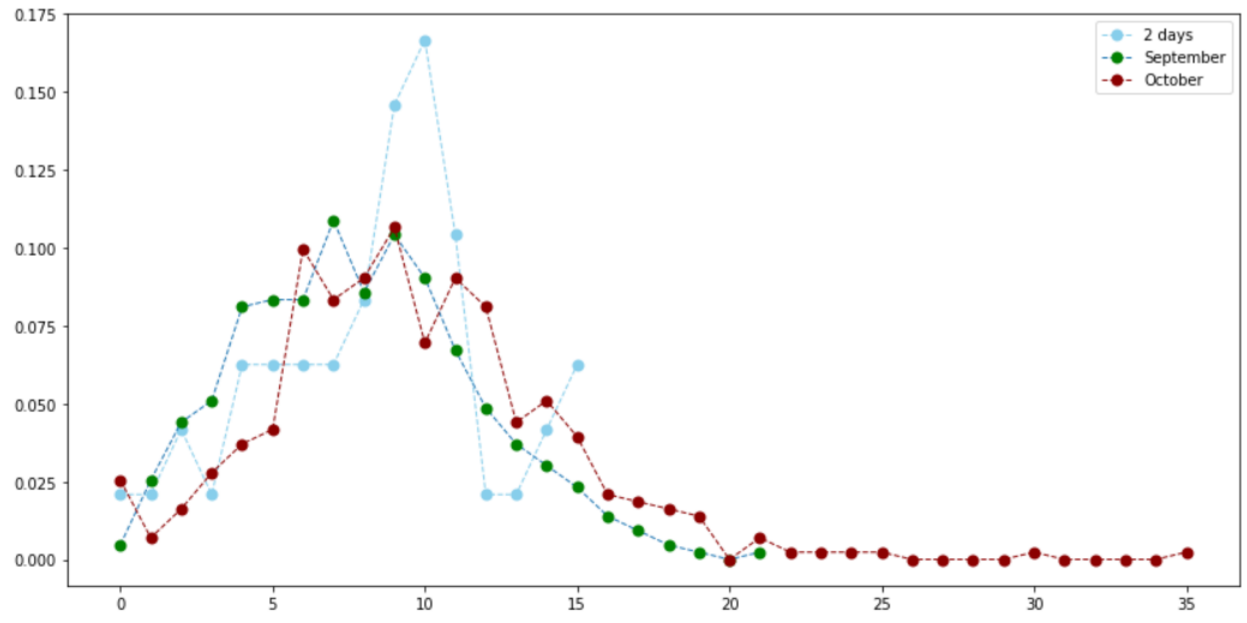
Running this code in conjunction with the rest of the file yielded the errors regarding our attempt to model the calls in September and the 2-day interval. We compared the pdfs of each of these periods and calculated the error of using both maximum likelihood estimation, and empirical estimation. The errors by both are

```
ML fitting errors
2 days: 0.43336098300266856
September: 0.5071641399725155

Nonparametric fitting errors
2 days: 0.5185185185185183
September: 0.4027777777777777
```

Additionally, we can view the pdfs that resulted from each of these methods. First, we have the maximum likelihood results in the following plot:



As well as the empirical results from the following plot:



(c) What is the method that performs best for the 2-day data? What method is better for the September data? What does this suggest

about parametric against non-parametric estimation depending on the amount of data available?

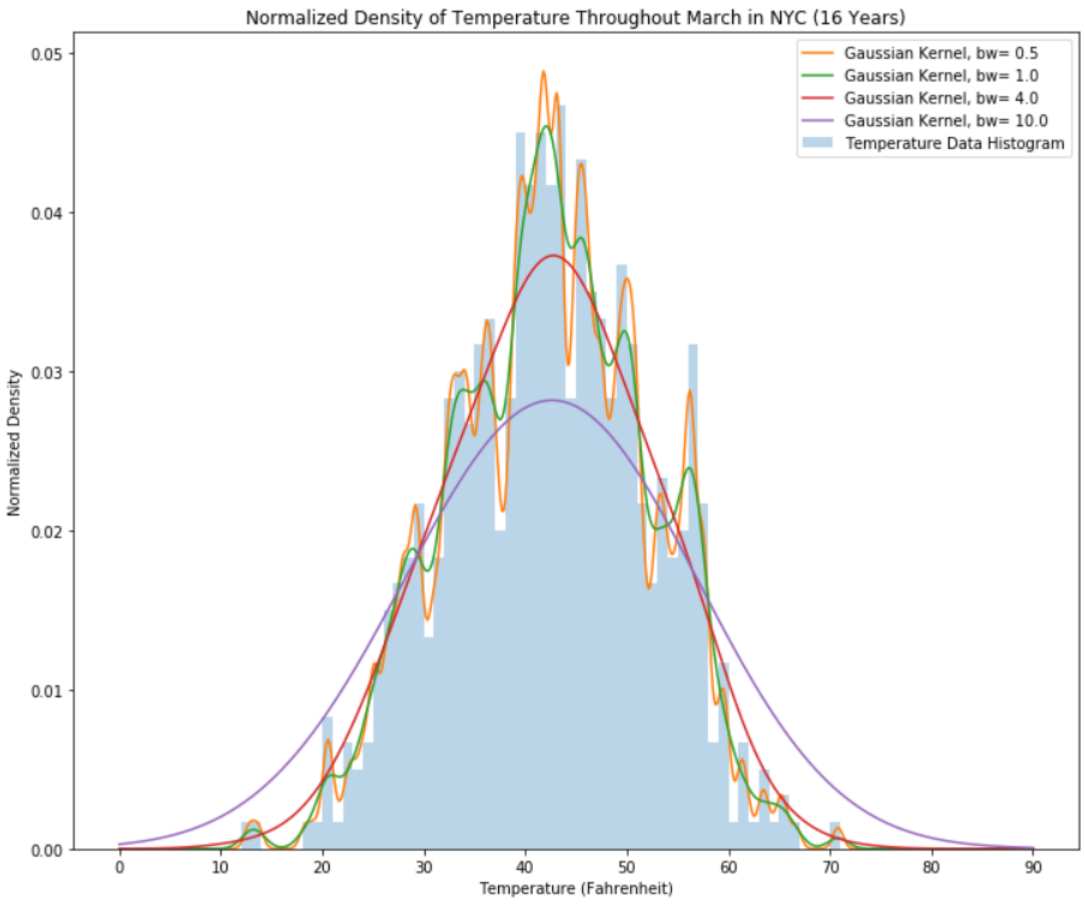
Maximum likelihood estimation appears to work best for the two day data, while the empirical/non-parametric method appears to work best for the September data. This seems to suggest that when we have small amounts of data, a parametric model works best, while larger amounts of data are better approximated with non-parametric models.

4. **(KDE) The *train.csv*, *val.csv* and *test.csv* files consist of daily average temperature of New York City in March over the course of 16 years. We are interested in the distribution of the average temperature.**

   (a) **Plot the kernel density estimator of the daily average temperature training data found in *train.csv*. You should use the Gaussian kernel with 4 different bandwidths: 0.5, 1, 4, 10. You should submit a single plot with the histogram of the data, and the 4 kernel density estimator with appropriate labels in the legend. [Note: You may make use of the functions in the *sklearn* library.]**

   Below, we have included the code used in this step, as well as the resulting plot,

```python
import pandas as pd
import matplotlib.pyplot as plt

f = pd.read_csv( r'C:\Users\Eric\Downloads\train.csv') ## Read in data

from sklearn.neighbors import KernelDensity
import numpy as np

t = list(f['AvgTemperature']) ## Extract data into an array
tt = [[i] for i in t]
T = np.array(tt)

X_plot = np.linspace(0, 90, 600)[:, np.newaxis]

fig, ax = plt.subplots(figsize=(12,10))
bins = np.linspace(0, 90, 91)
ax.hist(t, bins=bins, density=True, alpha=0.3, label = 'Temperature Data Histogram') ## Create normalized histogram
ax.set_ylabel('Normalized Density')
ax.set_xlabel('Temperature (Fahrenheit)')
ax.set_title('Normalized Density of Temperature Throughout March in NYC (16 Years)')
h = plt.xticks(np.arange(0, 91, step=10))  # Set label locations


for band in [0.5, 1.0, 4.0, 10.0]: # Use sklearn to create plots for each bandwidth
    kde = KernelDensity(kernel='gaussian', bandwidth=band).fit(T)
    log_dens = kde.score_samples(X_plot)
    ax.plot(X_plot[:, 0], np.exp(log_dens), label = 'Gaussian Kernel, bw= '+str(band))
    ax.legend()
```



(b) In this part we will use the validation data in *val.csv* to determine the optimal bandwidth setting. For each bandwidth in the range 0.5, 1, ..., 9.5, 10, perform the following calculation:

i. Compute the kernel density estimator on the training data using the given bandwidth.

The following is the code required to compute the kernel density estimators on the training data,

```
kernels = []
for band in [0.5*k for k in range(1,21)]:
    kde = KernelDensity(kernel='gaussian', bandwidth=band).fit(T)
    kernels.append(kde)
```

ii. Partition the interval [5, 80] into 15 intervals $I_1, ..., I_{15}$ of length 5.

This step was accomplished implicitly within other steps.

iii. Use the kernel density estimator to compute the probability of temperature lying in each interval.

The following is the code required to compute the probability of temperature lying in each interval. We created a function responsible for approximating the integral using trapezoids,

```
def trap_sum(a,b,N,kde):
    log_dens = kde.score_samples(np.linspace(a,b,N+1)[:, np.newaxis])
    y = np.exp(log_dens)
    y_right = y[1:]
    y_left = y[:-1]
    dx = (b - a)/N
    T = (dx/2) * np.sum(y_right + y_left)

    return(T)
```

```
prob_kernels = []

for ker in kernels:
    p_k = []

    for a in range(0,15):
        x = trap_sum(5+(a*5),10+(a*5),10000,ker)
        p_k.append(x)

    prob_kernels.append(p_k)
```

iv. Then use the data in *val.csv* to compute the probability temperature is in each interval.

The following is the code required to compute the compute the probability that resulting temperatures are in each interval. We used the empirical definition of probability,

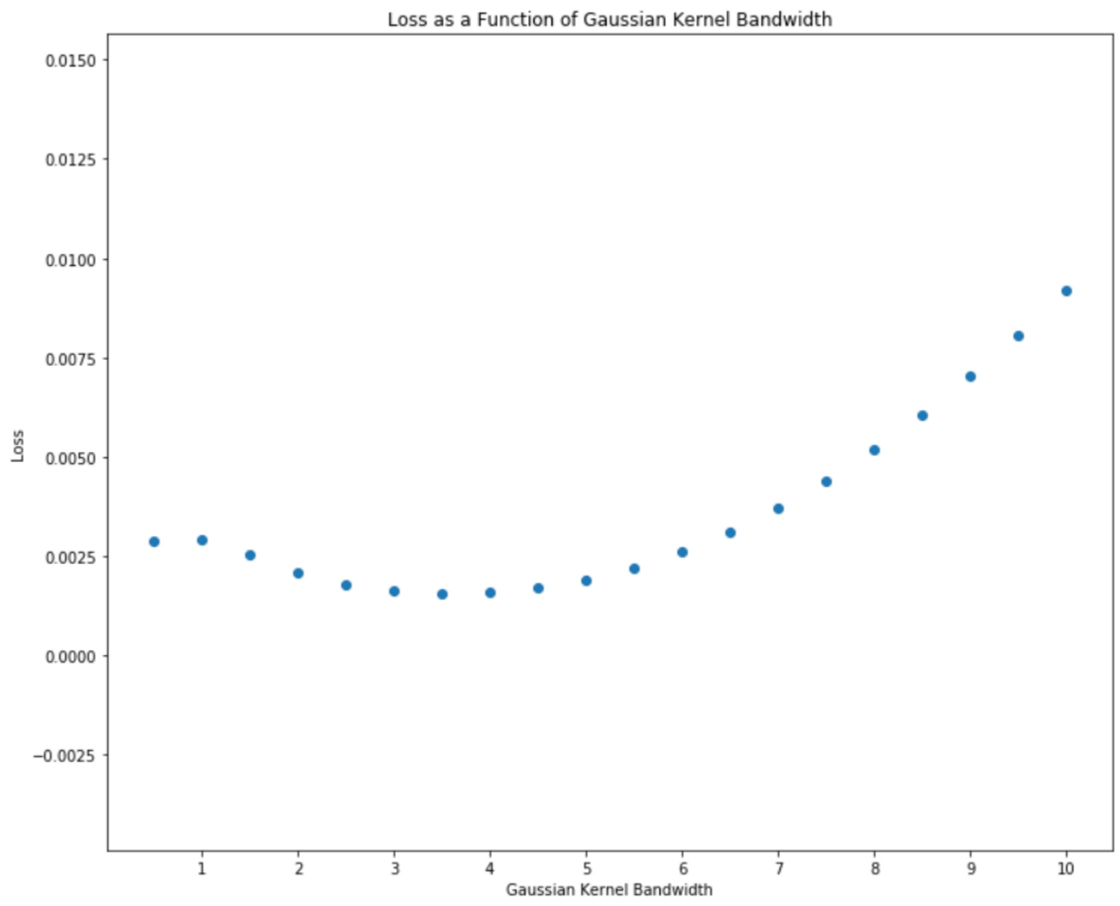
```
val = pd.read_csv( r'C:\Users\Eric\Downloads\val.csv')
counts = []
for k in range(0,15):
    v = len(val[(val['AvgTemperature'] >= 5+(5*k)) & (val['AvgTemperature'] < 10+(5*k))])
    counts.append(v)
counts = np.array(counts)/len(f['AvgTemperature'])
```

v. Finally, compute the loss defined to be the sum of the 15 square differences between the kernel density probabilities and the validation probabilities.

Submit a plot of loss vs. bandwidth, and also state the bandwidth that achieves the smallest loss.

The following is the code required to compute the loss between the kernel density probabilities and the validation probabilities. The code and plot of the losses for each kernel density estimation are included,

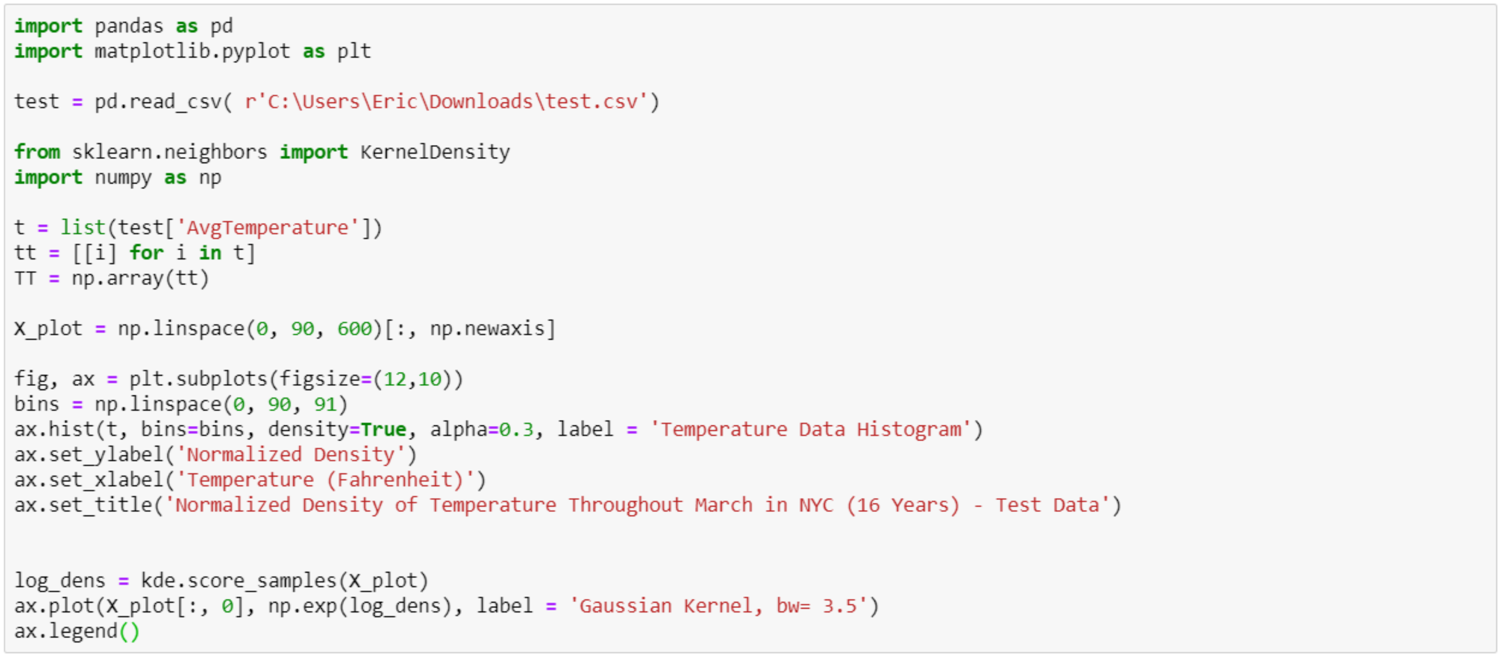
```
losses = []
for row in prob_kernels:
    loss = 0
    loss_vec = np.array(row) - np.array(counts)
    for e in loss_vec:
        loss += e**2
    losses.append(loss)
```



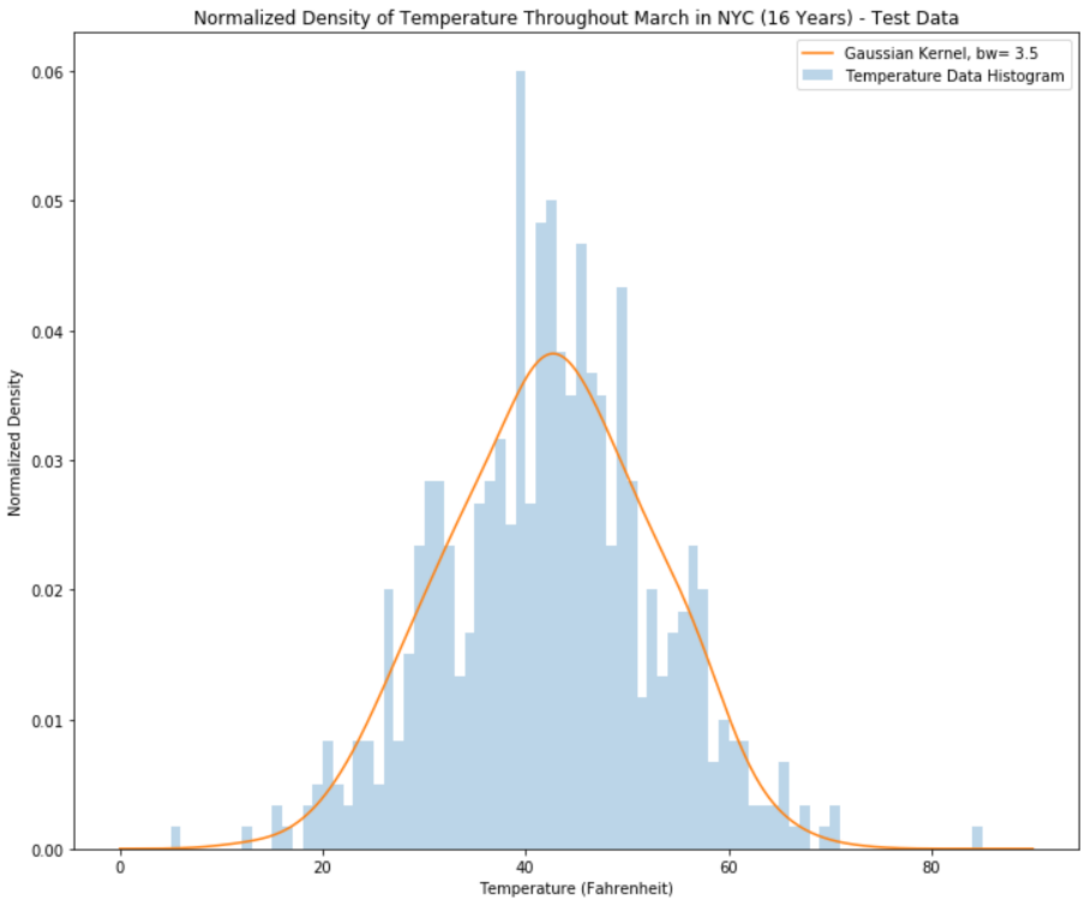
It is clear that the optimal bandwidth associated with the training data is 3.5, because this value for the bandwidth generates the lowest loss.

(c) Using the optimal bandwidth from the previous part, overlay the kernel density estimator (computed from the training data) on top of the histogram for the test data.

The following is the code and resulting plot of the kernel density estimator computed from the training data, with a bandwidth of 3.5, overlaid on the histogram of the test data,



```
import pandas as pd
import matplotlib.pyplot as plt

test = pd.read_csv( r'C:\Users\Eric\Downloads\test.csv')

from sklearn.neighbors import KernelDensity
import numpy as np

t = list(test['AvgTemperature'])
tt = [[i] for i in t]
TT = np.array(tt)

X_plot = np.linspace(0, 90, 600)[:, np.newaxis]

fig, ax = plt.subplots(figsize=(12,10))
bins = np.linspace(0, 90, 91)
ax.hist(t, bins=bins, density=True, alpha=0.3, label = 'Temperature Data Histogram')
ax.set_ylabel('Normalized Density')
ax.set_xlabel('Temperature (Fahrenheit)')
ax.set_title('Normalized Density of Temperature Throughout March in NYC (16 Years) - Test Data')


log_dens = kde.score_samples(X_plot)
ax.plot(X_plot[:, 0], np.exp(log_dens), label = 'Gaussian Kernel, bw= 3.5')
ax.legend()
```


Normalized Density of Temperature Throughout March in NYC (16 Years) - Test Data
Gaussian Kernel, bw= 3.5
Temperature Data Histogram
Normalized Density
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0
20
40
60
80
Temperature (Fahrenheit)