CSE512 Spring 2021 - Machine Learning - Homework 1

Your Name: Sagar Onkar Toshniwal

Solar ID: 113260061

NetID email address: sagaronkar.toshniwal@stonybrook.edu

Names of people whom you discussed the homework with: None but I have taken reference from Google, Stackoverflow, etc.

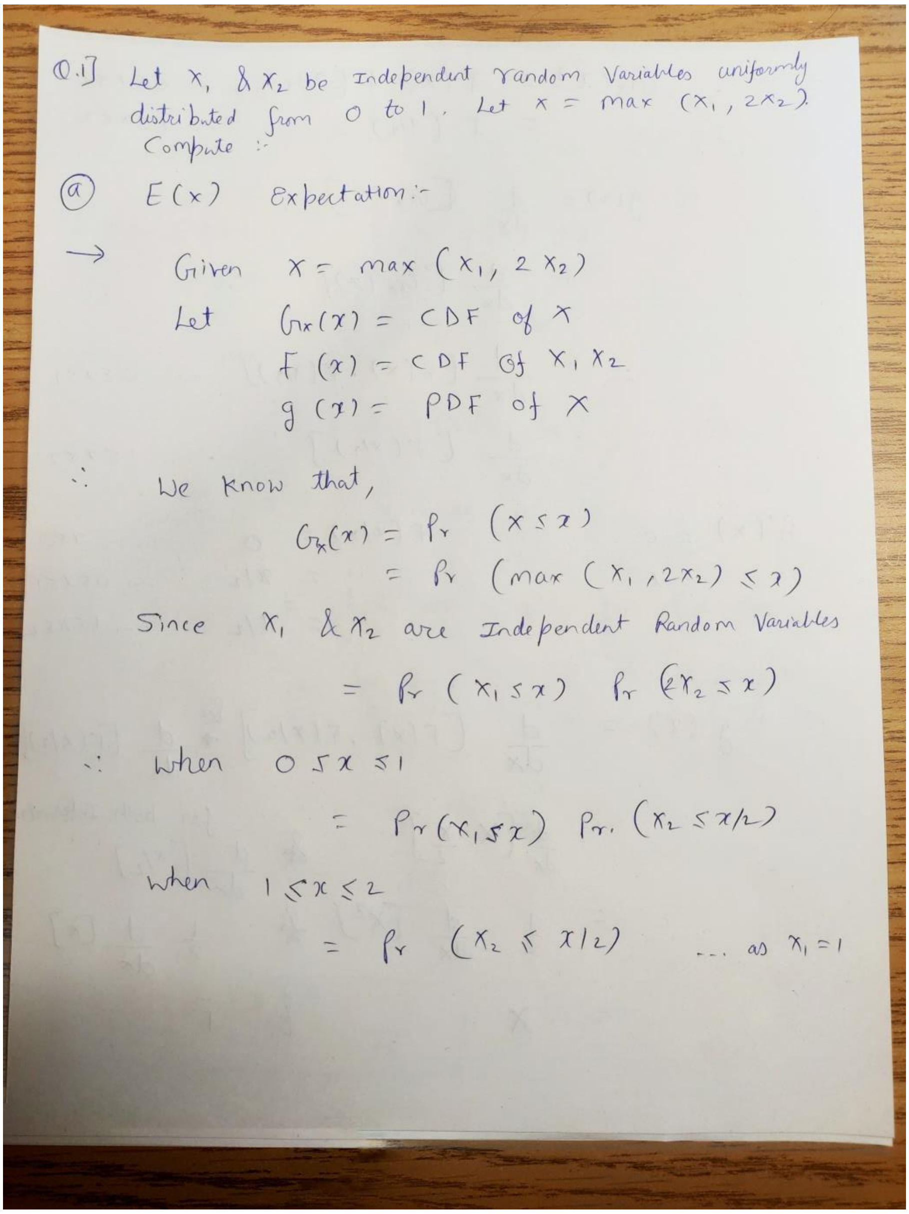
Question 1 – Probability (30 points)

Let X1 and X2 be independent continuous random variables uniformly distributed from 0 to 1. Let X = max(X1; 2X2). Compute:

1. The expectation E(X)

2. The variance V ar(X)

3. The covariance: Cov(X;X1).



Q.1] Let X_1 & X_2 be Independent random Variables uniformly distributed from 0 to 1. Let $X = \max(X_1, 2X_2)$.
Compute :-

ⓐ $E(X)$ Expectation :-

→ Given $X = \max(X_1, 2X_2)$

Let $G_X(x)$ = CDF of X
$F(x)$ = CDF of X_1, X_2
$g(x)$ = PDF of X

∵ We know that,

$$G_X(x) = P_r(X \le x)$$
$$= P_r(\max(X_1, 2X_2) \le x)$$

Since X_1 & X_2 are Independent Random Variables

$$= P_r(X_1 \le x)\ P_r(2X_2 \le x)$$

∴ when $0 \le x \le 1$

$$= P_r(X_1 \le x)\ P_r.(X_2 \le x/2)$$

when $1 \le x \le 2$

$$= P_r(X_2 \le x/2)$$ ... as $X_1 = 1$

$\therefore \quad G_X(x) = F(x) \cdot F(x/2) \quad \ldots 0 \le x \le 1$

$\qquad\qquad = F(x/2) \quad \ldots 1 \le x \le 2$

$\therefore \quad g(x) = \frac{d}{dx}\left[G_X(x)\right]$

$\qquad = \frac{d}{dx}\left[G_X(x)\right] \quad \ldots$

$\qquad = \frac{d}{dx}\left[F(x) \cdot F(x/2)\right] \quad \ldots 0 \le x \le 1$

$\qquad \frac{d}{dx}\left[F(x/2)\right] \quad \ldots 1 \le x \le 2$

$F(x) = 0 \qquad\qquad F(x/2) = 0 \quad \ldots x \le 0$

$\quad = x \qquad\qquad\qquad = x/2 \quad \ldots 0 \le x \le 1$

$\quad = 1 \qquad\qquad\qquad = x/2 \quad \ldots 1 \le x \le 2$

$\therefore \quad g(x) = \frac{d}{dx}\left[F(x) \cdot F(x/2)\right] \;\&\; \frac{d}{dx}\left[F(x/2)\right]$

$\ldots$ for both Intervals.

$\therefore \qquad = \frac{d}{dx}\left[x \cdot \frac{x}{2}\right] \quad \& \quad \frac{d}{dx}\left[x/2\right]$

$\qquad = \frac{1}{2} \frac{d}{dx}\left[x^2\right] \quad \& \quad \frac{1}{2} \frac{d}{dx}\left[x\right]$

$\qquad = x \quad \& \quad 1$

$$g(x) = x \quad \cdots \quad 0 \leq x \leq 1$$
$$= \frac{1}{2} \quad \cdots \quad 1 \leq x \leq 2$$

$\therefore$ (a) $E(x) = \int_{-\infty}^{\infty} x \cdot g(x)\, dx$

$$= \int_{-\infty}^{\infty} x \cdot g(x)\, dx$$

$$= \int_{0}^{1} x \cdot g(x) dx + \int_{1}^{2} x \cdot g(x)\, dx$$

$$= \int_{0}^{1} x \cdot x\, dx + \int_{1}^{2} x \cdot \frac{1}{2}\, dx$$

$$= \int_{0}^{1} x^2\, dx + \frac{1}{2} \int_{1}^{2} x\, dx$$

$$= \left[\frac{x^3}{3}\right]_0^1 + \frac{1}{2}\left[\frac{x^2}{2}\right]_1^2$$

$$= \frac{1}{3} + \frac{3}{4}$$

$$E(x) = \frac{13}{12}$$

$$\therefore \boxed{E(x) = \frac{13}{12}}$$

(b) Var (x)

$\rightarrow$ $Var(x) = E[x^2] - [E(x)]^2$

$\because E[x^2] = \int_{-\infty}^{\infty} x^2 \cdot g(x)\, dx$

$= \int_0^1 x^2 \cdot x\, dx + \int_1^2 x^2 \cdot \frac{1}{2}\, dx$

$= \int_0^1 x^3\, dx + \frac{1}{2}\int_1^2 x^2\, dx$

$= \left[\frac{x^4}{4}\right]_0^1 + \frac{1}{2} \cdot \frac{1}{3}\left[x^3\right]_1^2$

$= \frac{1}{4} + \frac{7}{6} = \frac{34}{24}$

$= \frac{17}{12}$

$\therefore Var(x) = \frac{17}{12} - \left(\frac{13}{12}\right)^2$

$= \frac{17}{12} - \frac{169}{144}$

$$\boxed{Var(x) = \frac{35}{144}}$$

③ $Cov(X, X_1)$

$\rightarrow$ $cov(X, X_1) = Cov(X_1, \max(X_1, X_2))$

$= cov(X_1, X_1)\ Pr(X_1 = \max(X_1, X_2))$

$+$

$cov(X_1, X_2)\ Pr\ (X = \max(X_1 X_2))$

$\because$ Since X_1 & X_2 are Independent.

$\therefore$ $Cov(X_1, X_2)$ must be 0

$= Cov(X_1, X_1)\ Pr\ (X_1 > X_2)$

$= Var(X_1)\ Pr.\ (X_1 > X_2)$

Since $cov(X_1 X_1) == Var(X_1)$

$= Var(X_1)\ [(1 - Pr\ X_1 \leq x)]$

$= Var(X_1)\ [\int_0^1 1 - x\,dx)]$

$Var(X_1) = E(X_1^2) - [E(X_1)]^2$

$= \int_0^1 x^2 dx - \left[\int_0^1 x\,dx\right]^2$

$= \left[\frac{x^3}{3}\right]_0^1 - \left\{\left[\frac{x^2}{2}\right]_0^1\right\}^2$

$$= \frac{1}{3} - \left(\frac{1}{2}\right)^2$$

$$= \frac{1}{3} - \frac{1}{4}$$

$$= \frac{1}{12}$$

$$\therefore \ cov(x, x_1) = Var(x_1)\left[\int_0^1 1-x \, dx\right]$$

$$= \frac{1}{12}\left[\int_0^1 1 \, dx - \int_0^1 x \, dx\right]$$

$$= \frac{1}{12}\left[1 - \left[\frac{x^2}{2}\right]_0^1\right]$$

$$= \frac{1}{12}\left[1 - \frac{1}{2}\right]$$

$$= \frac{1}{12} \times \frac{1}{2}$$

$$= \cancel{\frac{1}{24}} \ \frac{1}{24}$$

$$\boxed{\therefore \ cov(x, x_1) = \frac{1}{24}}$$

1. E(X) = 13/12
2. Var(X) = 35/144
3. Covv(X,X1) = 1/24

Answer for Question 2.2 :-

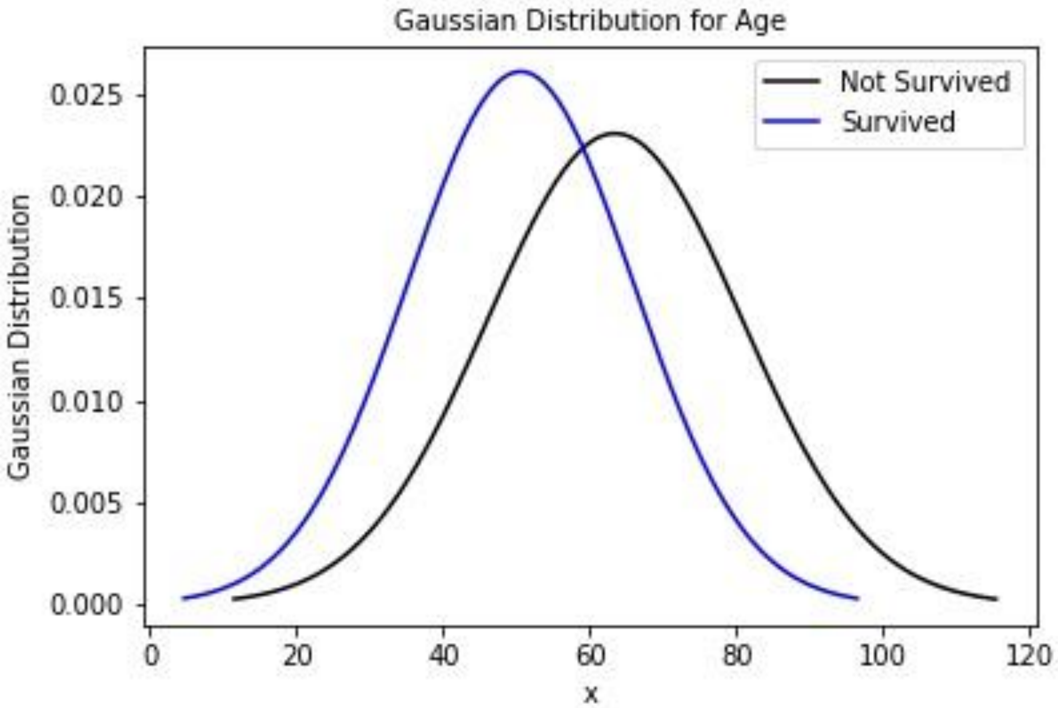
(a) (5 points) Report the values of mu0, var0, mu1, var1.

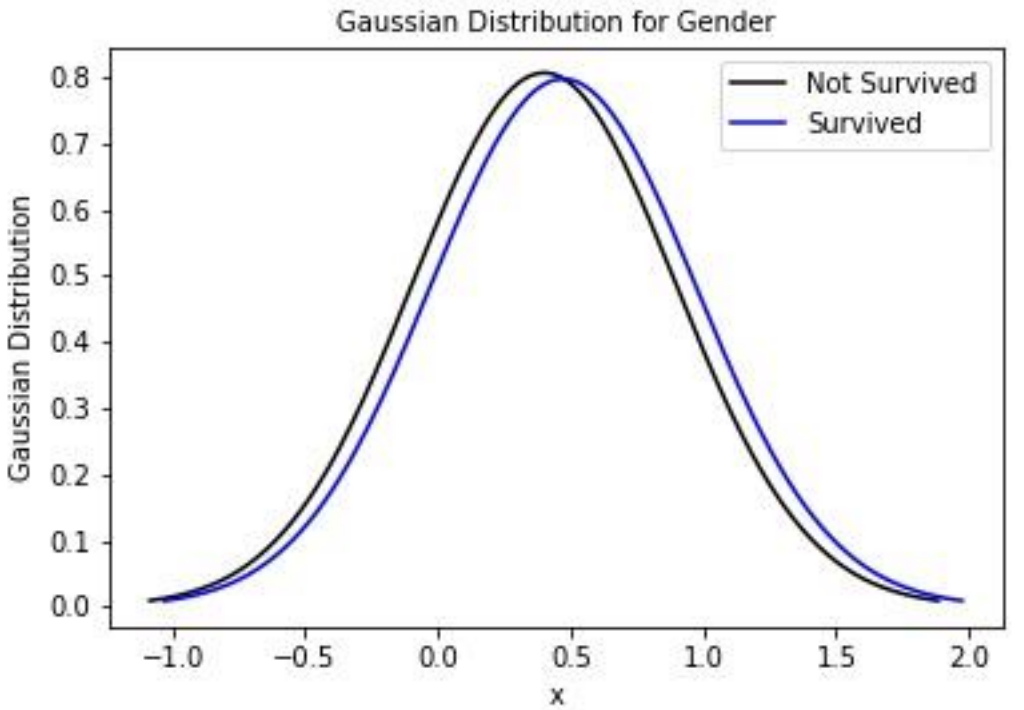
```
[54]: [mu0, var0, mu1, var1] = get_mean_and_variance(X, y)

      print ("Mu0 = ",mu0)
      print ("Var0 = ",var0)
      print ("Mu1 = ",mu1)
      print ("Var1 = ",var1)
```

```
Mu0 =  [63.38  0.4 ]
Var0 =  [2.99587347e+02 2.44897959e-01]
Mu1 =  [50.51685393  0.47191011]
Var1 =  [233.97994033   0.25061893]
```

Mu0 = [63.38 0.4]
Var0 = [2.99587347e+02 2.44897959e-01]
Mu1 = [50.51685393 0.47191011]
Var1 = [233.97994033 0.25061893]

(b) (5 points) For each feature j, plot the Gaussian distribution with mean mu0[j] and variance var0[j] in black color. On the same graph, plot the Gaussian distribution with mean mu1[j] and variance var1[j] in blue. You can use Python packages matplotlib and scipy.





(c) (5 points) Is it a good idea to approximate gender by a Gaussian distribution? Why or why not?

→ No it is not a good idea to approximate Gender by a Gaussian distribution for the following reason :-

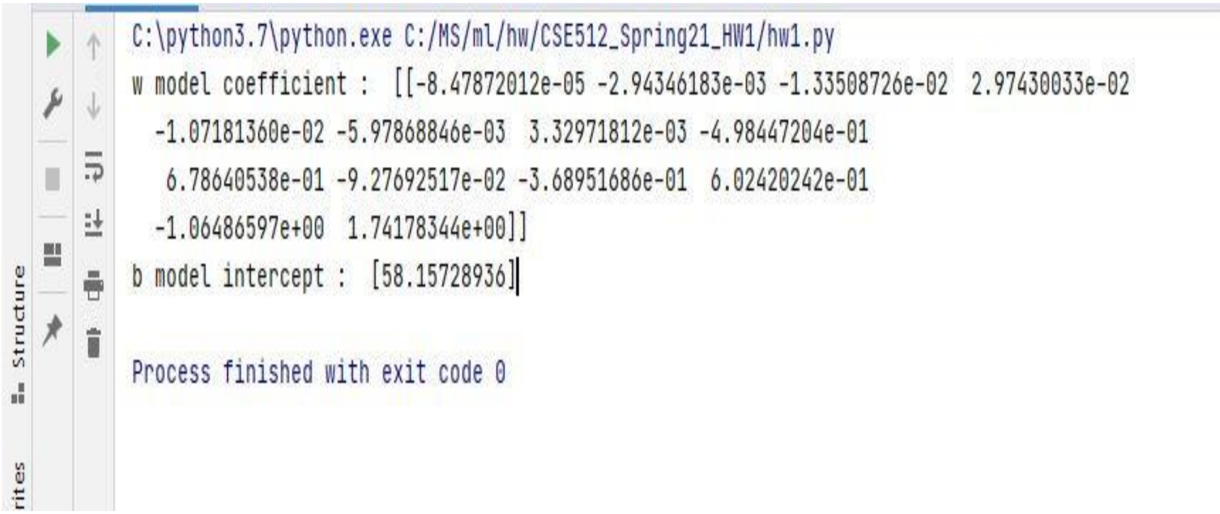
- Gaussian distribution is a type of continuous probability distribution for a real-valued random variable.
- Gender has limited outcomes and is discrete in nature. Gender is similar to a model which has the set of possible outcomes of any single experiment that asks a yes–no question.
- For such discrete data and possible outcomes, **Binomial distribution** is perfectly suited.

Python code for reference :-

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import norm
from sklearn.linear_model import LinearRegression
from sklearn import metrics

def get_mean_and_variance(X, y) :
    data = np.append(X,y,1)
    data_0 = data[data[:, -1] == 0, :-1]
    data_1 = data[data[:, -1] == 1, :-1]

    mu0 = np.mean(data_0, axis =0)
    var0 = np.var(data_0, axis =0)
    mu1 = np.mean(data_1, axis =0)
    var1 = np.var(data_1, axis =0)
    return mu0,var0,mu1,var1

df = pd.read_csv("covid19_metadata.csv")
y = df.iloc[:,-1:]
y = y.replace(to_replace = ['Y','N'],value = [1,0])
X = df.iloc[:,:-1]
X.gender = X.gender.replace(to_replace = ['F','M'],value = [1,0])

[mu0, var0, mu1, var1] = get_mean_and_variance(X, y)

print ("mu0 = ",mu0)
print ("var0 = ",var0)
print ("mu1 = ",mu1)
print ("var1 = ",var1)

sigma0 = np.sqrt(var0)
sigma1 = np.sqrt(var1)

fig, (ax1,ax2) = plt.subplots (1,2, figsize = (12,9))
xpts0 = np.linspace(mu0[0] - 3*sigma0[0], mu0[0] + 3*sigma0[0], 200)
xpts1 = np.linspace(mu1[0] - 3*sigma1[0], mu1[0] + 3*sigma1[0], 200)

ax1.plot(xpts0, norm.pdf(xpts0, mu0[0], sigma0[0]),color = 'black')
ax1.plot(xpts1, norm.pdf(xpts1, mu1[0], sigma1[0]),color = 'blue')
ax1.set_title('Gaussian Distribution for Age',fontsize=10)

ax1.set(xlabel='x', ylabel='Gaussian Distribution')
ax1.legend(['Not Survived', 'Survived'])

xpts0 = np.linspace(mu0[1] - 3*sigma0[1], mu0[1] + 3*sigma0[1], 200)
xpts1 = np.linspace(mu1[1] - 3*sigma1[1], mu1[1] + 3*sigma1[1], 200)
ax2.plot(xpts0, norm.pdf(xpts0, mu0[1], sigma0[1]),color = 'black')
ax2.plot(xpts1, norm.pdf(xpts1, mu1[1], sigma1[1]),color = 'blue')
ax2.set_title('Gaussian Distribution for Gender',fontsize=10)

ax2.set(xlabel='x', ylabel='Gaussian Distribution')
ax2.legend(['Not Survived', 'Survived'])
plt.show()
```

Answer for Question 3.2 :-

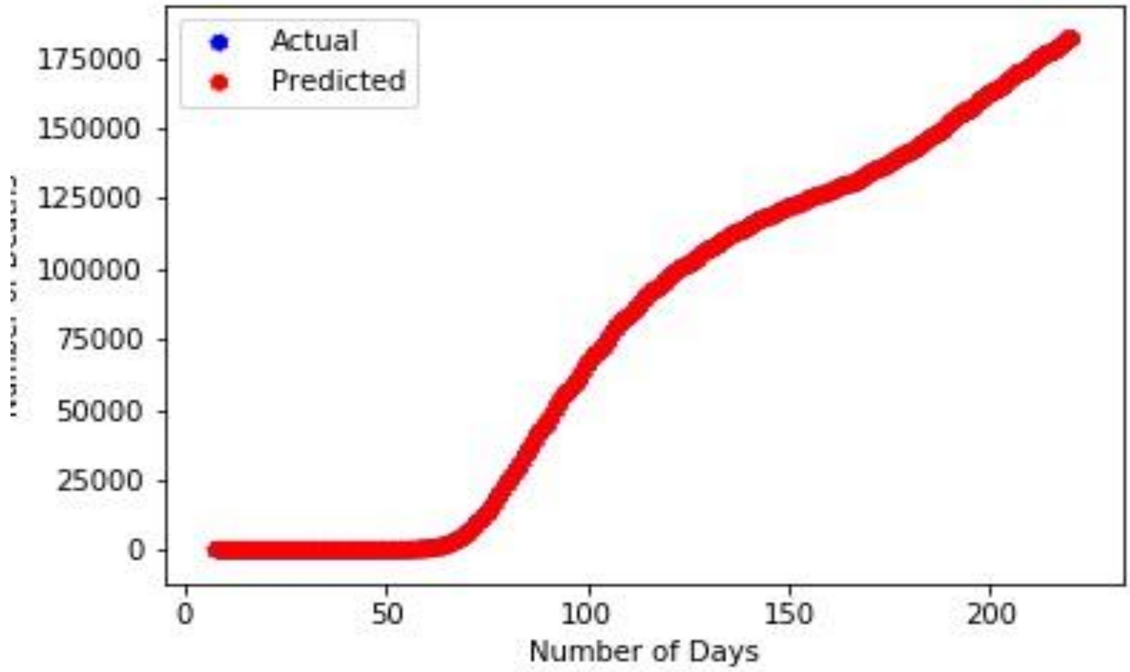
(a) (5 points) Report the learned parameters: the weights w and the intercept term b.

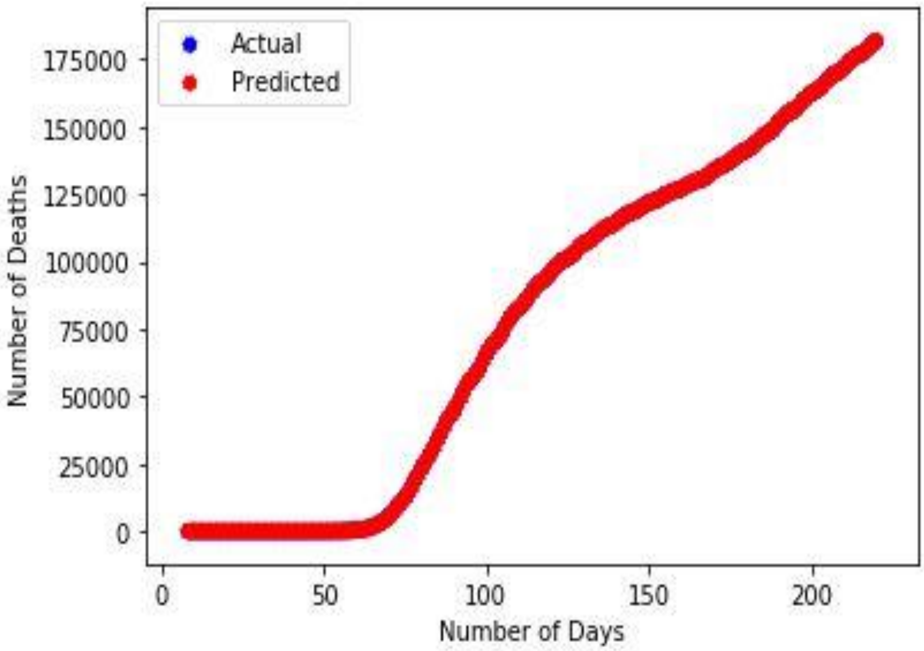


w model coefficient : **[[-8.47872012e-05**
-2.94346183e-03
-1.33508726e-02
2.97430033e-02
-1.07181360e-02
-5.97868846e-03
3.32971812e-03
-4.98447204e-01
6.78640538e-01
-9.27692517e-02
-3.68951686e-01
6.02420242e-01
-1.06486597e+00
1.74178344e+00]]

b model intercept : **[58.15728936]**

(b)(5 points) Visualize the actual and predicted death values yt and ^yt (for 8 <= t <= n). Display yt as a function of t and ^yt as a function of t on the same graph. You can use the library matplotlib.pyplot to plot.





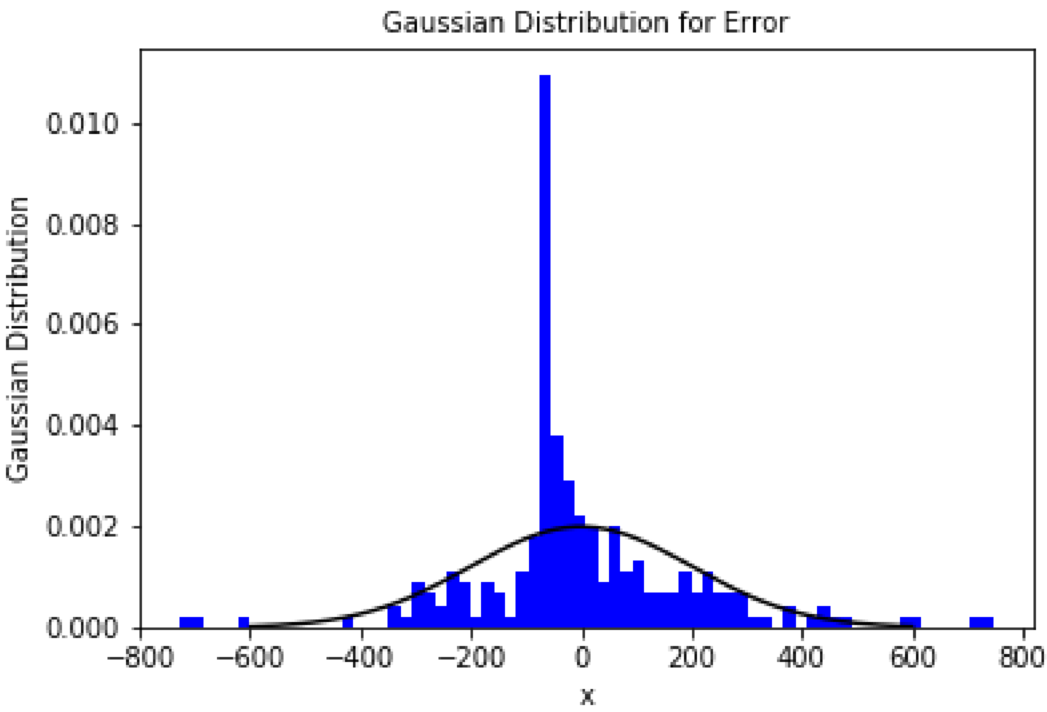
(c) (5 points) Use a Gaussian to approximate the distribution of the errors yt - ^yt (for 8 <= t <= n). Report the mean and variance of this Gaussian.

```
[89]: va = np.var(error,ddof = 1)
      std = np.sqrt(va)

      print ("Mean for Gaussian distribution of error : ", me)
      print ("Variance for Gaussian distribution of error : ", va)
```

```
Mean for Gaussian distribution of error :  -1.2600558051323527e-11
Variance for Gaussian distribution of error :  39949.11578698032
```

Mean for Gaussian distribution of error : **-1.2600558051323527e-11**
Variance for Gaussian distribution of error : **39949.11578698032**

(d) (5 points) Use matplotlib.pyplot.hist to plot the distribution of yt - ^yt (for 8 <= t <= n). On the same plot, plot the Gaussian function that approximates this distribution. Is Gaussian a good approximation for the distribution of the errors?



- For approximation of error under Gaussian distribution, each error point should be independent and identically distributed which basically means that the error associated with a time value at a particular time point should be independent of error associated with a different time value at a different time point.
- If this case is satisfied then the Gaussian distribution is a good approximation for the distribution of the errors
- Because in the current given model, the error in the current point is dependent on the previous time point value, the IID (Independent and Identically distributed) assumption is violated
- Hence in our Case, it is not a good Idea to approximate distribution of error using Gaussian function

Python code for reference :-

```
def learn_reg_params(x, y) :
    y_copy = y[0, 7:]
    y_copy = np.atleast_2d(y_copy).T
    X = np.zeros((x.shape[1] - 7, 14))
    for t in range(7, x.shape[1]):
        X[t - 7,] = np.concatenate([x[0, t - 7:t], y[0, t - 7:t]])
    model = LinearRegression()
    model.fit(X, y_copy)
    y_pred = model.predict(X)
    print("Learned parameter w = ", model.coef_)
    print("Learned parameter b = ", model.intercept_)
    error = y_copy - y_pred
    me = np.mean(error)
    var = np.var(error, ddof=1)
    std = np.sqrt(var)

    fig, (ax3, ax4, ax5) = plt.subplots(1, 3, figsize=(12, 9))
    t = np.arange(8, 221)
    ax3.scatter(t, y_copy, marker='o', color='blue', linestyle=':')
    ax3.scatter(t, y_pred, marker='o', color='red', linestyle=':')
    ax3.set(xlabel='days', ylabel='Death')
    ax3.legend(['Actual', 'Predicted'])
    ax3.set_title("Actual and Predicted Death Values")

    xpts0 = np.linspace(me - 3 * std, me + 3 * std, 200)
    ax4.plot(xpts0, norm.pdf(xpts0, me, std), color='black')

    ax4.set_title('Gaussian Distribution for Errors', fontsize=10)
    ax4.set(xlabel='X', ylabel='Error')

    xpts0 = np.linspace(me - 3 * std, me + 3 * std, 200)

    ax5.plot(xpts0, norm.pdf(xpts0, me, std), color='black')
    ax5.hist(error, bins=70, color='blue', density=True)
    ax5.set_title('Gaussian Distribution and Histogram plot for Error',
fontsize=10)

    ax5.set(xlabel='X', ylabel='Error')

    plt.show()
    print("w model coefficient : ", model.coef_)
    print("b model intercept : ", model.intercept_)
    return model.coef_,model.intercept_


df1 = pd.read_csv('covid19_time_series.csv', ',')
data1 = df1.to_numpy()
x = data1[:1,1:]
y = data1[1:,1:]
w,b = learn_reg_params(x, y)
```