CGS698C, Assignment 2

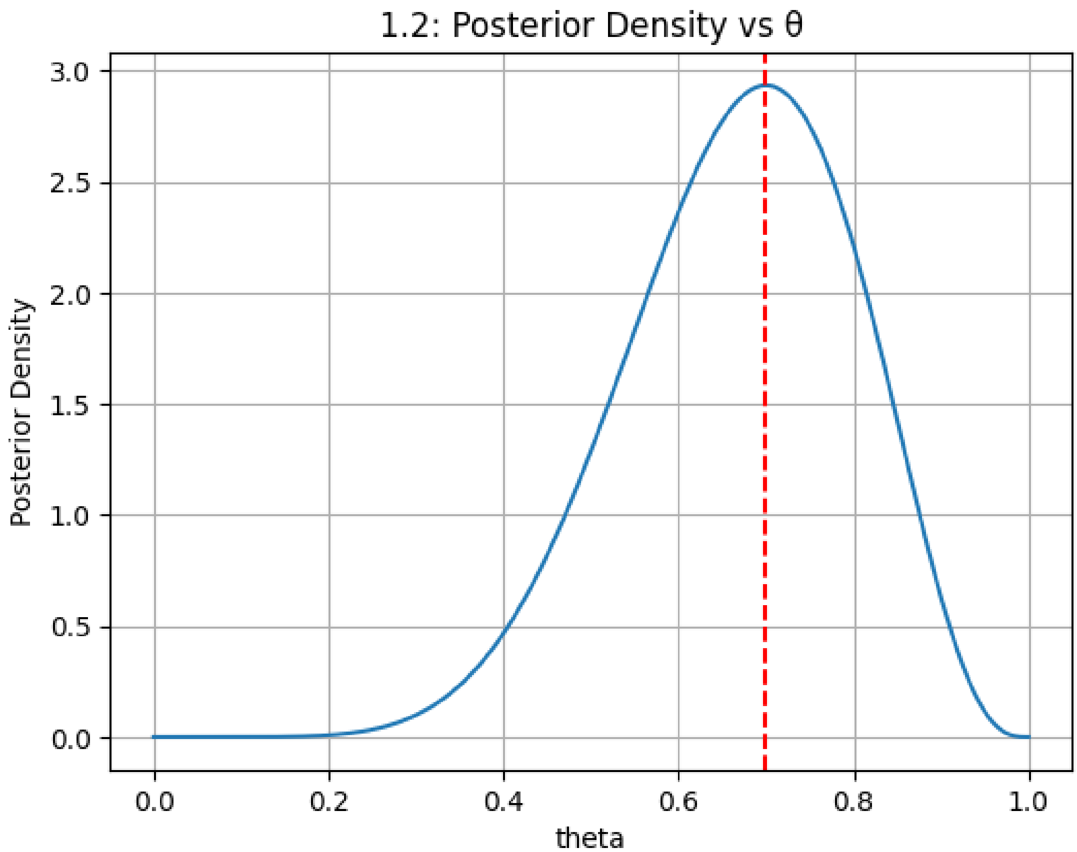
Arpita Chaurasia

Part 1: ***A Simple Binomial Model***

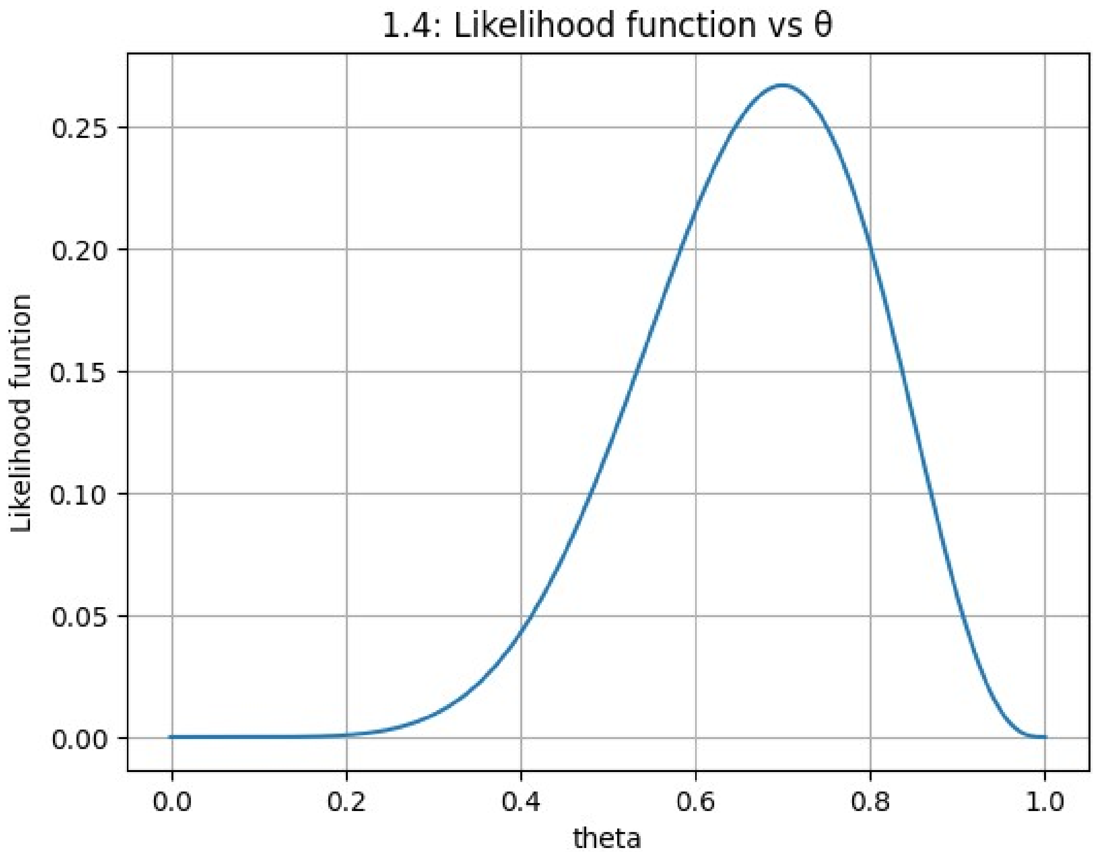
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import binom

n=10
def likelihood_func(y, n, theta):
  return binom.pmf(y, n, theta)

def prior_dist(theta):
  if theta < 0 or theta > 1: return 0
  return 1

def posterior_dist(y, theta, marginal_likelihood):
  return likelihood_func(y, n,
theta)*prior_dist(theta)/marginal_likelihood

y=7
theta_datas = [0.75, 0.25, 1.0]
marginal_likelihood = 1/11

for theta in theta_datas:
   posterior_density = posterior_dist(y, theta, marginal_likelihood)
   print (f"1.1 (a),(b),(c) theta = {theta}, posterior density:
{posterior_density}")
print(" ")

#1.2
theta_values = np.linspace(0, 1, 1000)
posterior_densities = [posterior_dist(y, theta, marginal_likelihood)
for theta in theta_values]

plt.plot(theta_values, posterior_densities)
plt.grid(True)
plt.xlabel('theta')
plt.ylabel('Posterior Density')
plt.title('1.2: Posterior Density vs θ')

#1.3
max_theta = theta_values[np.argmax(posterior_densities)]
plt.axvline(x=max_theta, color='r', linestyle='--')
plt.show()
print(" ")
```

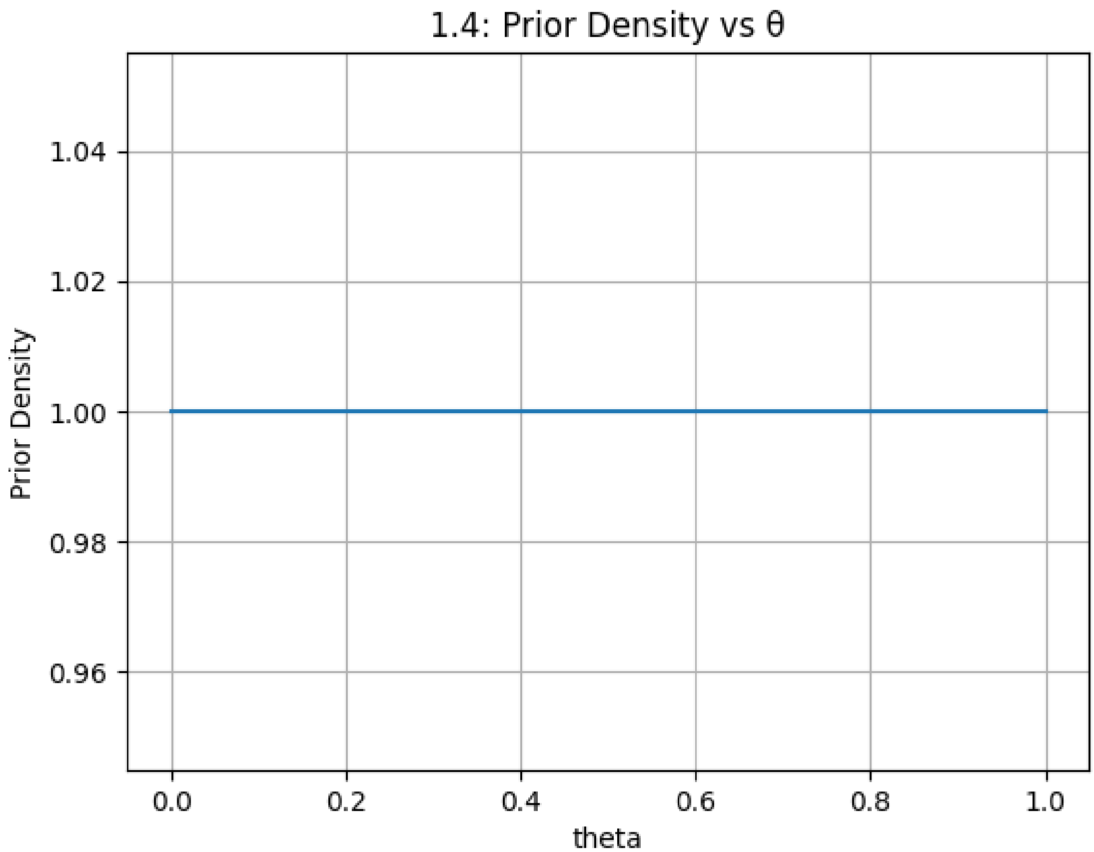
```python
print(f"1.3 : theta for maximum posterior density: {max_theta}")
print(" ")

#1.4
likelihoods = [likelihood_func(y, n, theta) for theta in theta_values]
plt.plot(theta_values, likelihoods)
plt.grid(True)
plt.xlabel('theta')
plt.ylabel('Likelihood funtion')
plt.title('1.4: Likelihood function vs θ')
plt.show()
print(" ")

prior_distribution = [prior_dist(theta) for theta in theta_values]
plt.plot(theta_values, prior_distribution)
plt.grid(True)
plt.xlabel('theta')
plt.ylabel('Prior Density')
plt.title('1.4: Prior Density vs θ')
plt.show()
print(" ")

posterior_densities = [posterior_dist(y, theta, marginal_likelihood)
for theta in theta_values]
plt.plot(theta_values, posterior_densities)
plt.grid(True)
plt.xlabel('theta')
plt.ylabel('Posterior Density')
plt.title('1.4: Posterior Density vs θ')
plt.show()
print(" ")
```

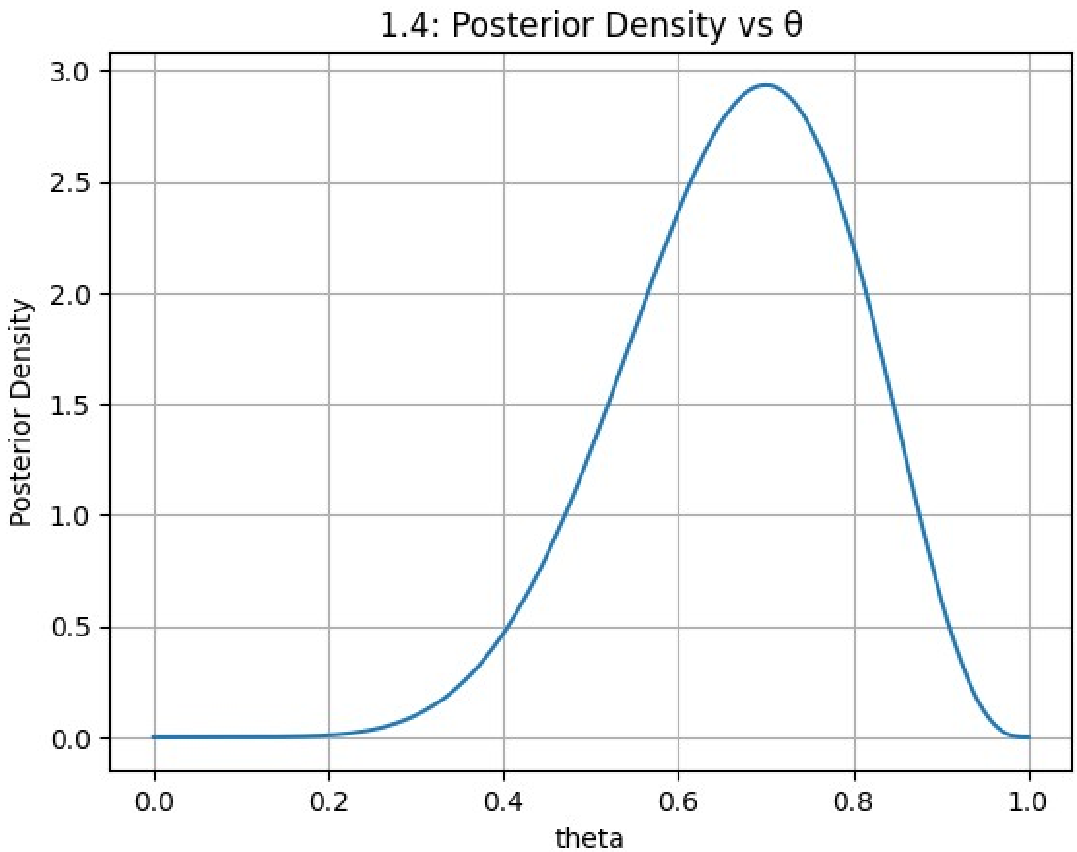
```
1.1 (a),(b),(c) theta = 0.75, posterior density: 2.7531051635742174
1.1 (a),(b),(c) theta = 0.25, posterior density: 0.03398895263671874
1.1 (a),(b),(c) theta = 1.0, posterior density: 0.0
```




```
1.3 : theta for maximum posterior density: 0.6996996996996997
```


1.4: Likelihood function vs θ
Likelihood funtion
0.00
0.05
0.10
0.15
0.20
0.25
0.0
0.2
0.4
0.6
0.8
1.0
theta


1.4: Prior Density vs θ
Prior Density
theta
1.04
1.02
1.00
0.98
0.96
0.0
0.2
0.4
0.6
0.8
1.0



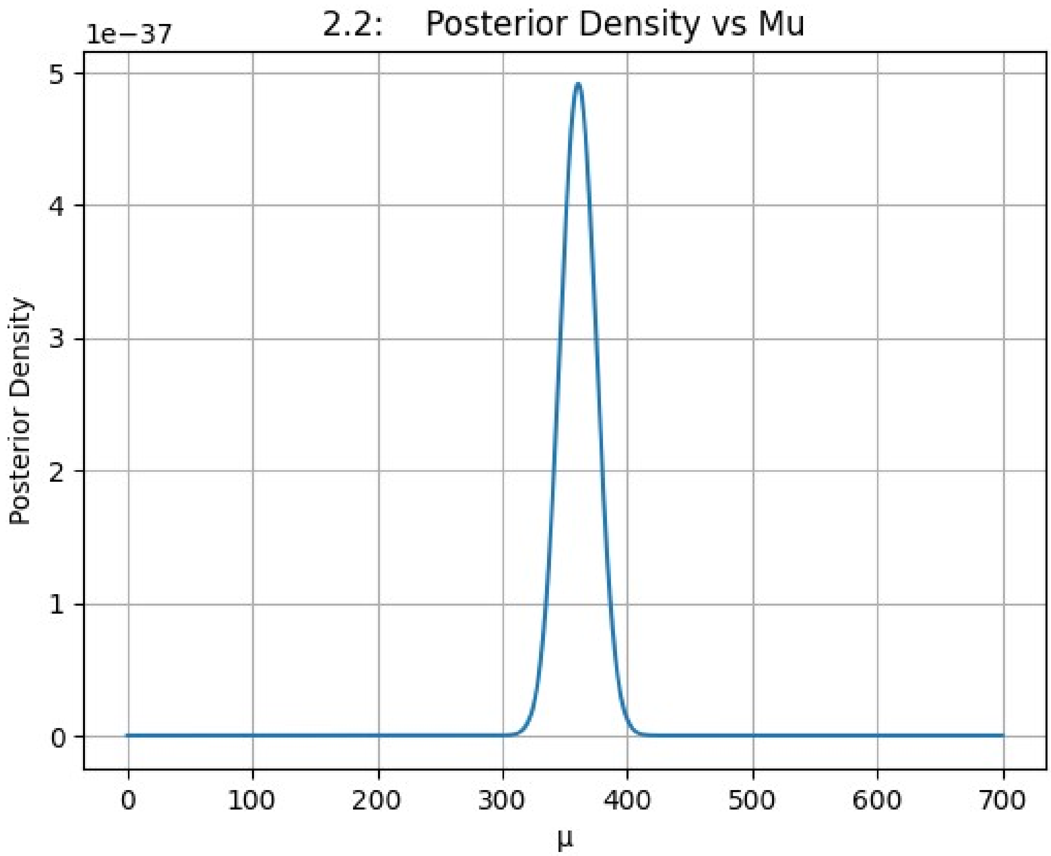
Part 2: ***A Gaussian Model of Reading***

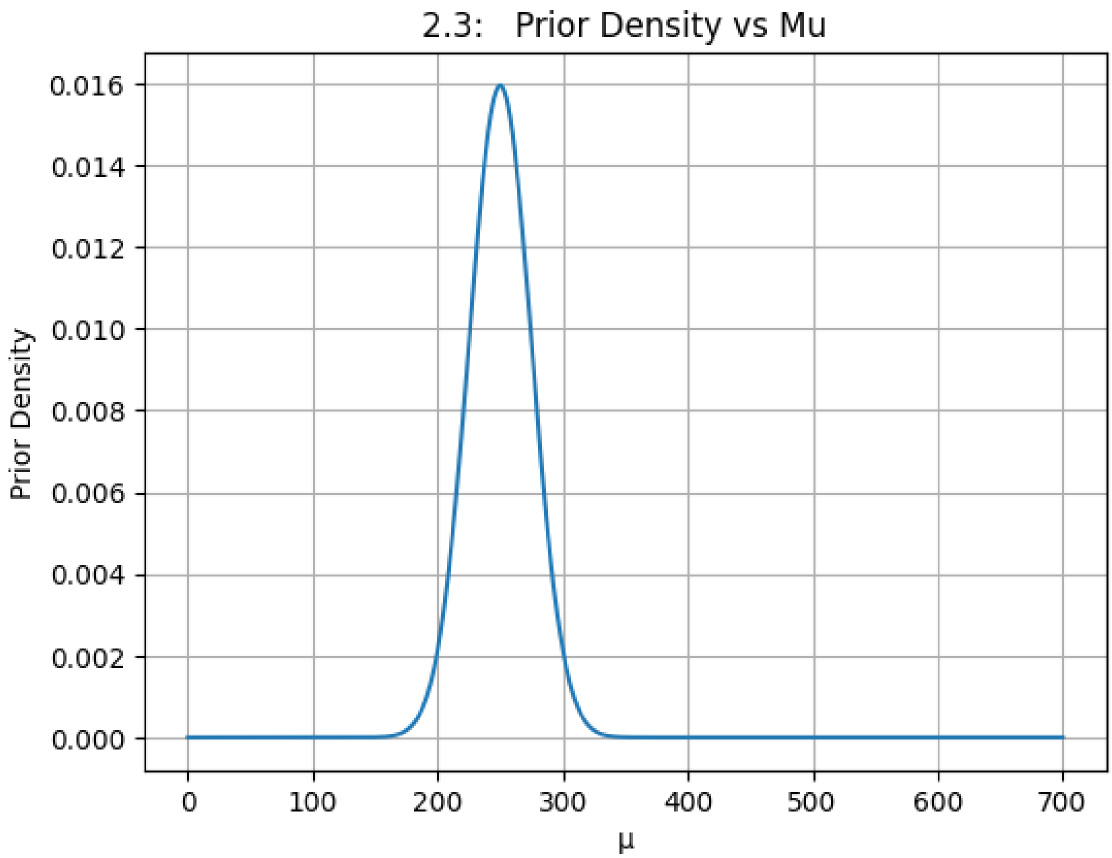
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

def likelihood_func(mu, sigma, y):
    return (1 / (sigma * np.sqrt(2 * np.pi)))** 8 * np.exp(-np.sum((y
- mu)** 2) / (2 * sigma**2))

def prior_dist(mu):
    mu_prior = (1 / (25 * np.sqrt(2 * np.pi))) * np.exp(-0.5 * ((mu -
250) / 25)**2)
    return mu_prior

def posterior_dist(mu, sigma, y):
    return likelihood_func(mu, sigma, y)*prior_dist(mu)

#2.1

sigma = 50
```

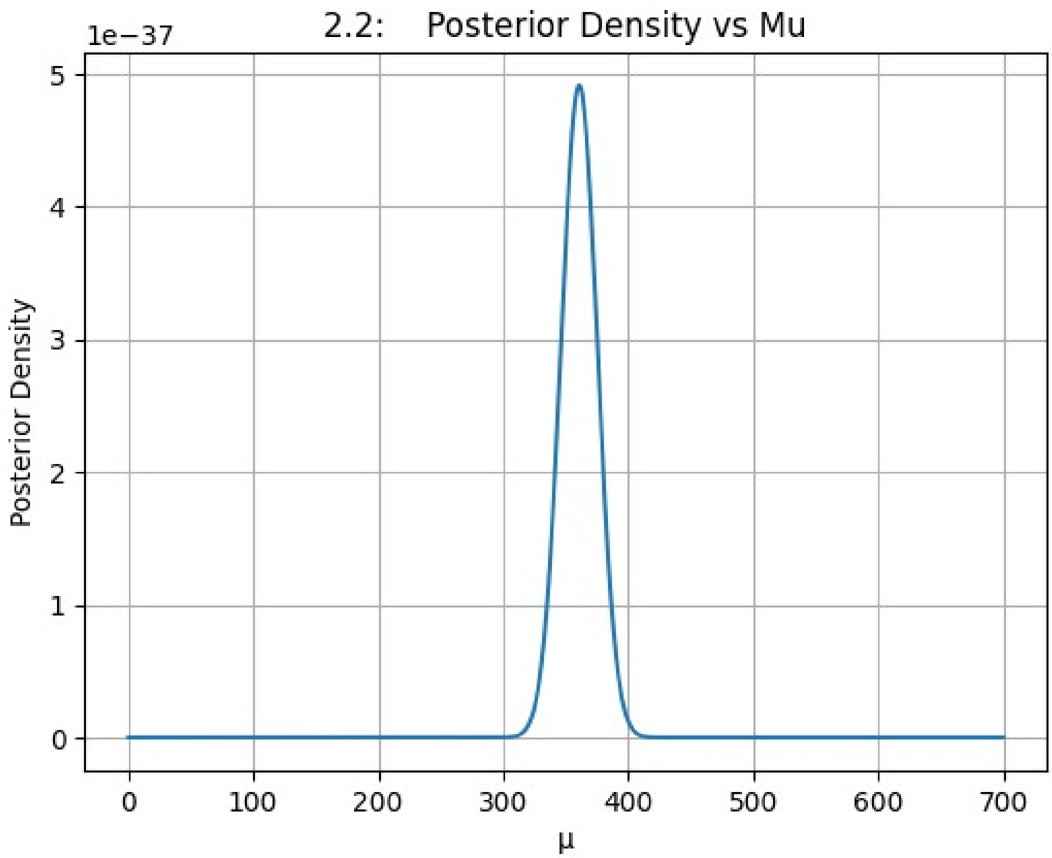
```
mu_datas = [300, 900, 50]
y = np.array([300, 270, 390, 450, 500, 290, 680, 450])

for mu in mu_datas:
        unnorm_posterior_density = posterior_dist(mu, sigma, y)
        print(f"2.1 (a),(b),(c) mu = {mu}, unnormalized posterior
density: {unnorm_posterior_density}")
print(" ")

#2.2

mu_values = np.linspace(0, 700, 1000)
posterior_densities = [posterior_dist(mu, sigma, y) for mu in
mu_values]

plt.plot(mu_values, posterior_densities)
plt.grid(True)
plt.xlabel('μ')
plt.ylabel('Posterior Density')
plt.title('2.2:    Posterior Density vs Mu')
plt.show()
print(" ")

#2.3
def prior_dist(mu):
   return (1 / (25 * np.sqrt(2 * np.pi))) * np.exp(-0.5 * ((mu - 250)
/ 25)**2)
prior_densities = [prior_dist(mu) for mu in mu_values]
plt.plot(mu_values, prior_densities)
plt.grid(True)
plt.xlabel('μ')
plt.ylabel('Prior Density')
plt.title('2.3:   Prior Density vs Mu')
plt.show()
print(" ")

posterior_densities = [posterior_dist(mu, sigma, y) for mu in
mu_values]
plt.plot(mu_values, posterior_densities)
plt.grid(True)
plt.xlabel('μ')
plt.ylabel('Posterior Density')
plt.title('2.2:    Posterior Density vs Mu')
#plt.tight_layout()
plt.show()
print(" ")
```

```
2.1 (a),(b),(c) mu = 300, unnormalized posterior density:
6.824247957486404e-41
2.1 (a),(b),(c) mu = 900, unnormalized posterior density: 0.0
```

```
2.1 (a),(b),(c) mu = 50, unnormalized posterior density:
9.691373559300646e-138
```




2.3: Prior Density vs Mu
Prior Density
0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
0
100
200
300
400
500
600
700
μ



Part 3: ***The Bayesian Learning***

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import poisson, gamma

n=2
k_datas = [25, 20, 23, 27]

#def posterior_dist(k, n):
#    posterior_dens = gamma.pdf(40+k, n)
#   return posterior_dens

for k in k_datas:
    n=n+1
    k = sum(k_datas)

day_5_k = (40+k)/n

day_5_posterior = posterior_dist(k,n)
```

```
print("Prior distribution parameters for day 5 (posterior of day 
4) :")
print("alpha:", 40+k)
print("beta:", n)

print(f"Predicted number of road accidents on day 5 (mean of 
posterior): {day_5_k}")
```

```
Prior distribution parameters for day 5 (posterior of day 4) :
alpha: 135
beta: 6
Predicted number of road accidents on day 5 (mean of posterior): 22.5
```

Answer 3.1

Prior parameters of day 5 will be the posterior parameters of day 4. Hence, alpha = 40 + sum of all 4 days' k = 40 + 95 = 135

Also beta for day 1 = 2, so beta for day 5 = 6.

Answer 3.2

Mean given by posterior distribution of day 5 is 22.5, so predicted number of road accidents on day 5 = 22 or 23.

Part 4: ***Model Building in Bayesian Framework***

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm, truncnorm
import pandas as pd

 #4.5.1 Null hypothesis model- posterior distribution for mu

url = 
"https://raw.githubusercontent.com/yadavhimanshu059/CGS698C/main/notes
/Module-2/recognition.csv"
data = pd.read_csv(url)
Tw = data['Tw']
Tnw = data['Tnw']

mean = 300
sigma = 60
delta = 0

def likelihood(mu, delta):
    likelihood_tw = norm.pdf(Tw, loc=mu, scale=sigma).prod()
    likelihood_tnw = norm.pdf(Tnw, loc=mu + delta, scale=sigma).prod()
    return likelihood_tw * likelihood_tnw

def prior_mu(mu):
```

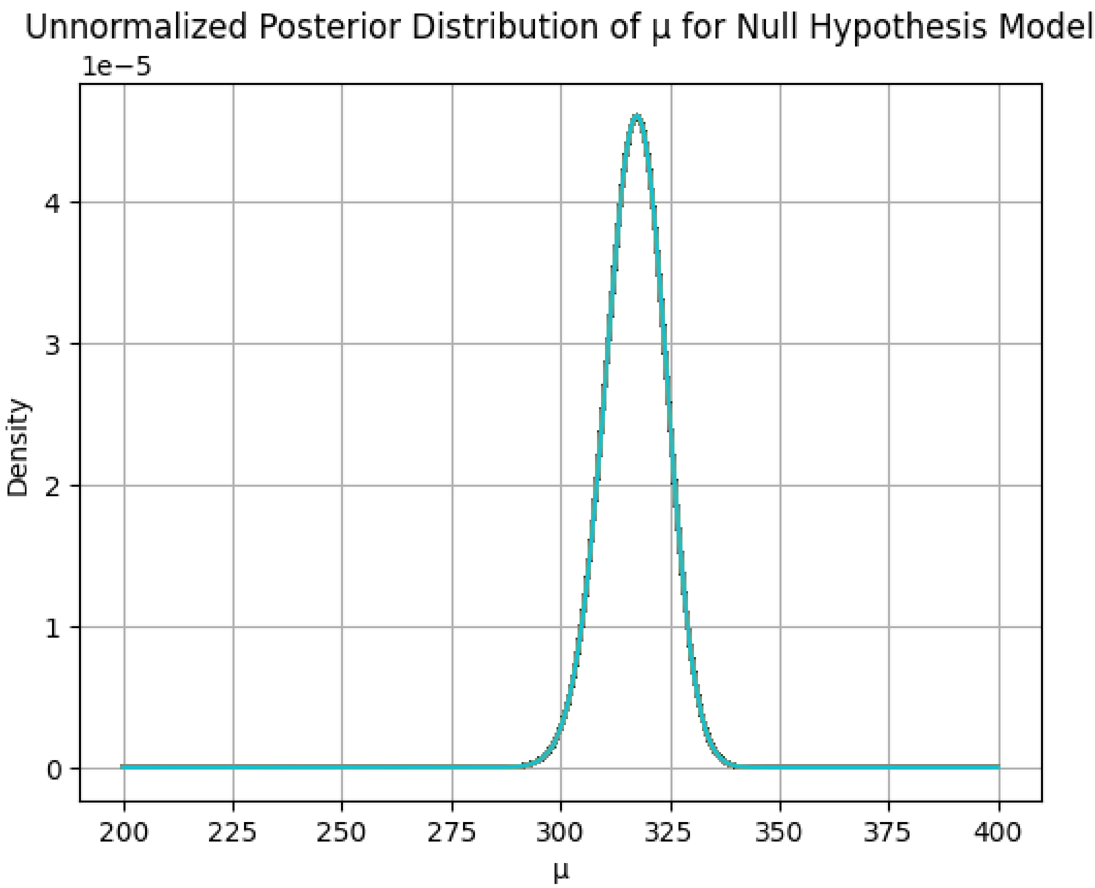
```
    return norm.pdf(mu, loc=300, scale=50)

def prior_delta(delta):
    return truncnorm.pdf(delta, a=0, b=np.inf, loc=0, scale=50)

mu_values = np.linspace(200, 400, 500)
delta_values = np.linspace(-100, 100, 500)

posterior_values = np.zeros((len(mu_values), len(delta_values)))
for i, mu in enumerate(mu_values):
    #for j, delta in enumerate(delta_values):
    for delta in delta_values:
        posterior_values[i] += likelihood(mu, delta) * prior_mu(mu) *
prior_delta(delta)

posterior_values = posterior_values / posterior_values.sum()

print("4.5.1")
plt.plot(mu_values, posterior_values, label='Unnormalized Posterior',
linestyle ='-')
print(" ")
plt.xlabel('μ')
plt.ylabel('Density')
plt.title('Unnormalized Posterior Distribution of μ for Null
Hypothesis Model')
plt.grid(True)
plt.show()
print(" ")

#4.5.2 Lexical access model - Prior predictions
n=1000
mu_samples = np.random.normal(mean, sigma, n)
delta_samples = np.random.normal(delta, sigma, n)

lex_nw_time = mu_samples + delta_samples + np.random.normal(0, sigma,
n)
lex_w_time = mu_samples + np.random.normal(0, sigma, n)

print("4.5.2")
plt.figure(figsize=(7, 6))
plt.hist(lex_nw_time, bins=30, alpha=0.5, label='Non-Word Recognition
Times', color='pink')
plt.hist(lex_w_time, bins=30, alpha=0.5, label='Word Recognition
Times', color='green')
plt.xlabel('Recognition times')
plt.ylabel('Frequency')
plt.title('Lexical-Access Model Prior Prediction')
plt.legend()
plt.grid(True)
plt.show()
```

```
print(" ")

#4.5.3 Null hypothesis model vs Lexical acess model

null_nw_time = np.random.normal(mean, sigma, n)
null_w_time = np.random.normal(mean, sigma, n)

print("4.5.3")
plt.figure(figsize=(6, 8))
plt.subplot(2, 1, 1)
plt.hist(null_w_time, bins=30, alpha=0.5, label='Word',
color='purple')
plt.hist(null_nw_time, bins=30, alpha=0.5, label='Non-Word',
color='orange')
plt.xlabel('Recognition Times')
plt.ylabel('Frequency')
plt.title('Null Hypothesis Model')
plt.legend()
plt.grid(True)
plt.show()
print(" ")

plt.figure(figsize=(6, 8))
plt.subplot(2, 1, 2)
plt.hist(lex_nw_time, bins=30, alpha=0.5, label='Non-Word Recognition
Times', color='pink')
plt.hist(lex_w_time, bins=30, alpha=0.5, label='Word Recognition
Times', color='green')
plt.xlabel('Recognition times')
plt.ylabel('Frequency')
plt.title('Lexical-Access Model Prior Prediction')
plt.legend()
plt.grid(True)
plt.show()
print(" ")
#4.5.4 Models' prior prediction comparison with observed data Tw and
Tnw

print("4.5.4")
plt.subplot(2, 2, 1)
plt.hist(null_w_time, bins=50, density=True, alpha=0.7,
color='skyblue', label='Prior')
plt.hist(Tw, bins=20, density=True ,alpha=0.3, color = 'gray',
label='Observed Data' )
plt.scatter(Tw, np.zeros_like(Tw), color='red', zorder=10)
plt.title('Null Model / Observed (Tw)')
plt.xlabel('Recognition Time')
plt.ylabel('Density')
plt.legend()
```

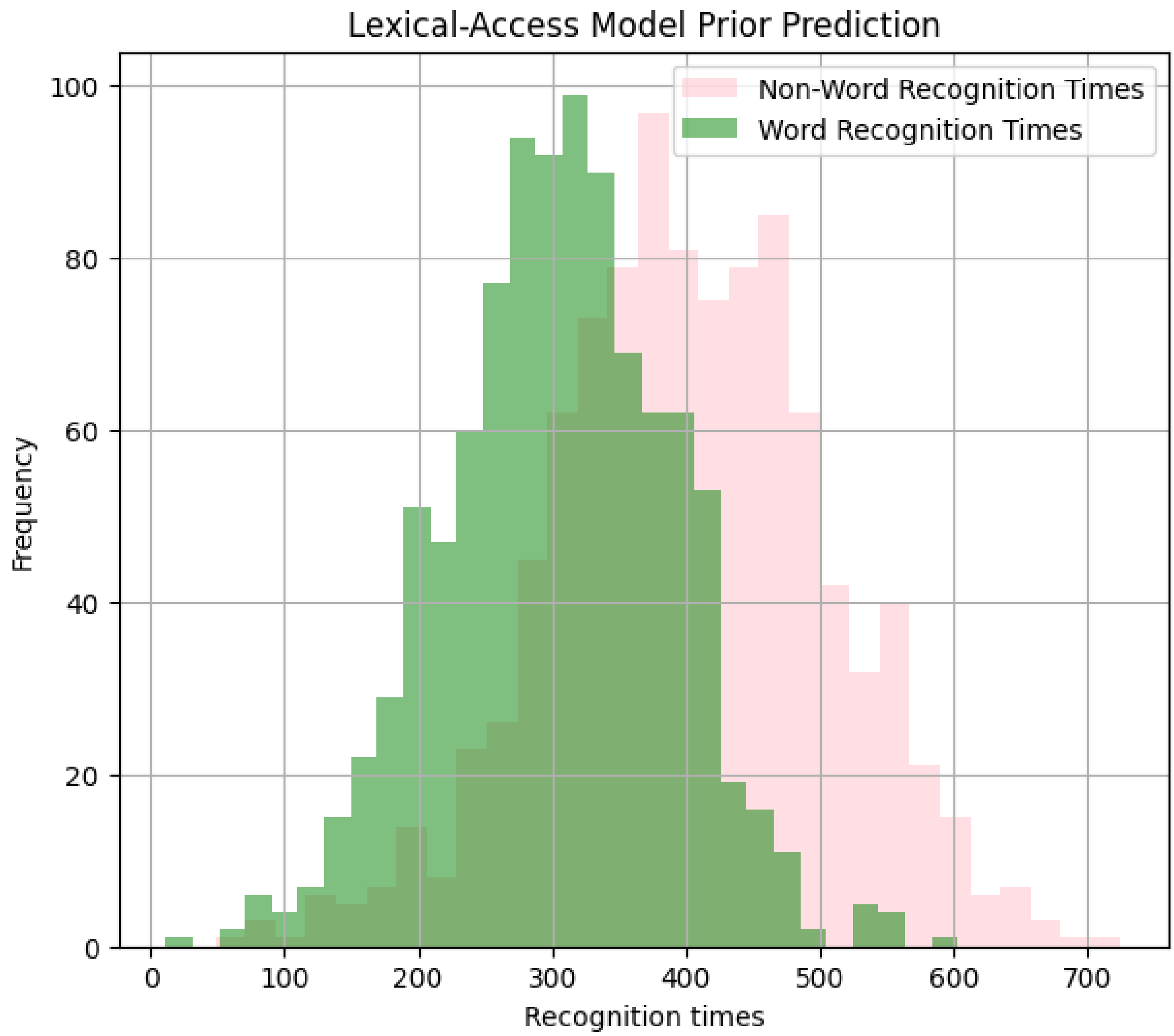
```
plt.subplot(2, 2, 2)
plt.hist(null_nw_time, bins=50, density=True, alpha=0.7,
color='pink', label='Prior')
plt.scatter(Tnw, np.zeros_like(Tnw), color='red', zorder=10)
plt.hist(Tnw, bins=20, density=True ,alpha=0.3, color = 'gray',
label='Observed Data' )
plt.title('Null Model / Observed (Tnw)')
plt.xlabel('Recognition Time')
plt.ylabel('Density')
plt.legend()

# Lexical-access model

plt.subplot(2, 2, 3)
plt.hist(lex_w_time, bins=50, density=True, alpha=0.7, color='coral',
label='Prior')
plt.hist(Tw, bins=15, density=True ,alpha=0.3, color = 'gray',
label='Observed Data' )
plt.scatter(Tw, np.zeros_like(Tw), color='red', zorder=10)
plt.title('Lexical Model / Observed (Tw)')
plt.xlabel('Recognition Time')
plt.ylabel('Density')
plt.legend()

plt.subplot(2, 2, 4)
plt.hist(lex_nw_time, bins=50, density=True, alpha=0.7,
color='lightgreen',
label='Prior')
plt.scatter(Tnw, np.zeros_like(Tnw), color='red', zorder=10)
plt.hist(Tnw, bins=15, density=True ,alpha=0.3, color = 'gray',
label='Observed Data' )
plt.title('Lexical Model / Observed (Tnw)')
plt.xlabel('Recognition Time')
plt.ylabel('Density')
plt.legend()
plt.tight_layout()
plt.show()

#4.5.5 Unnormalized posterior delta distribution
def unnormalized_posterior_lexical(delta, mu, Tw, Tnw, sigma):
    likelihood_tw = norm.pdf(Tw, loc=mu, scale=sigma).prod()
    likelihood_tnw = norm.pdf(Tnw, loc=mu + delta, scale=sigma).prod()
    return likelihood_tw * likelihood_tnw * prior_mu(mu) *
prior_delta(delta)

delta_values = np.linspace(0, 200, 1000)
mu_values = np.mean(mean)
posterior_delta_values = [unnormalized_posterior_lexical(delta,
mu_values, Tw, Tnw, sigma) for delta in delta_values]
print("4.5.5")
```

```
plt.plot(delta_values, posterior_delta_values)
plt.xlabel('δ')
plt.ylabel('Unnormalized Posterior')
plt.title('Unnormalized Posterior Distribution of δ (Lexical-Access
Model)')
plt.show()
```

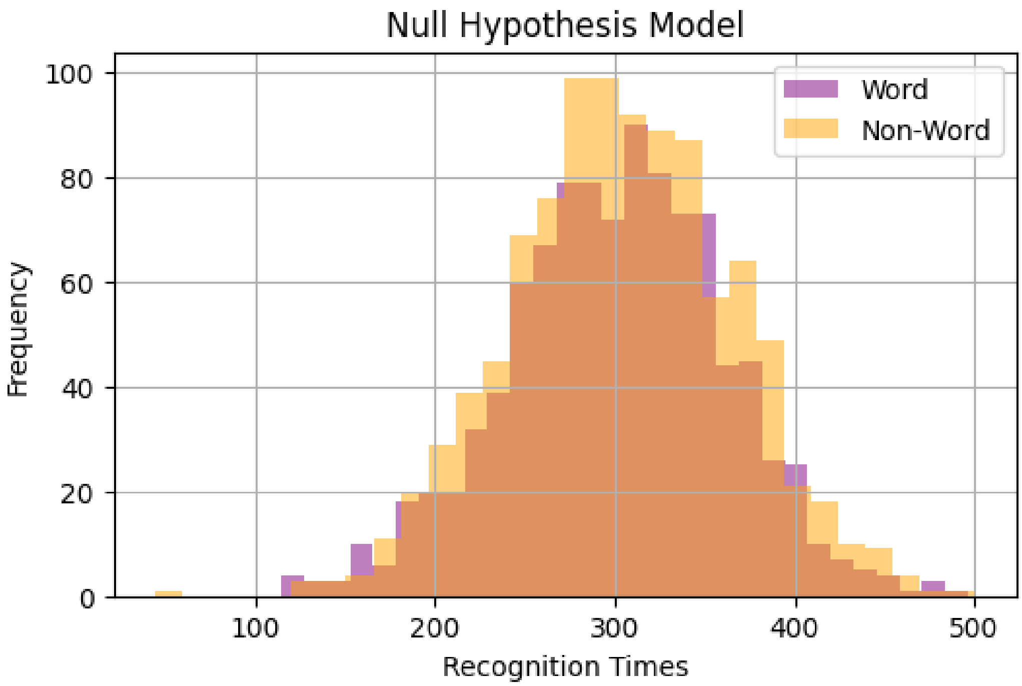
```
4.5.1
```

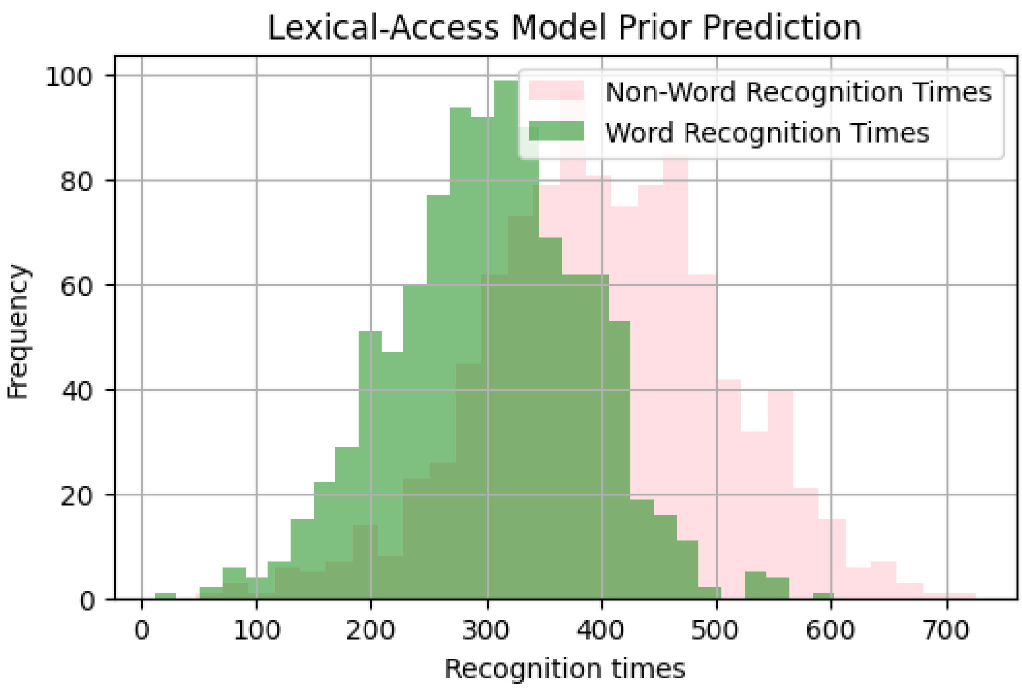



```
4.5.2
```

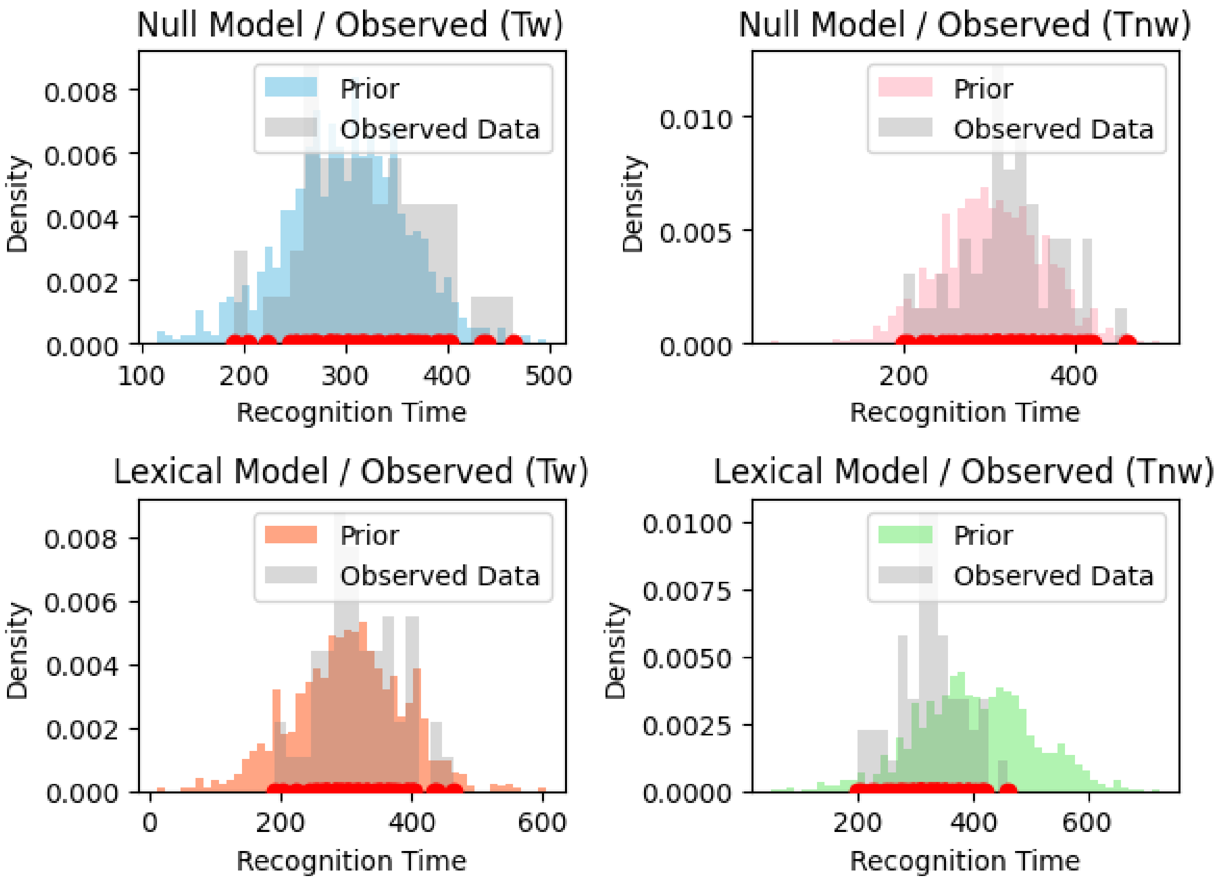

Lexical-Access Model Prior Prediction
Non-Word Recognition Times
Word Recognition Times
Frequency
Recognition times
0
20
40
60
80
100
0
100
200
300
400
500
600
700

```
4.5.3
```

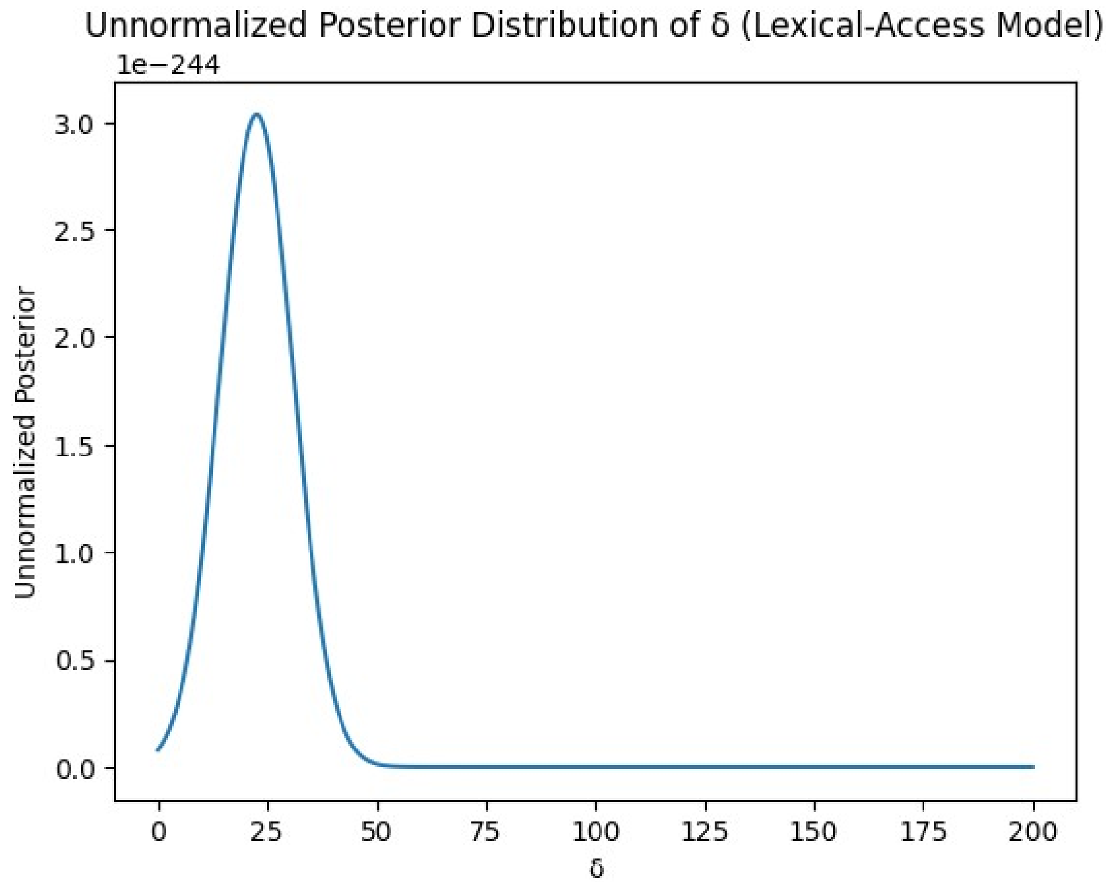

Null Hypothesis Model
Word
Non-Word
Frequency
Recognition Times
0
20
40
60
80
100
100
200
300
400
500


Lexical-Access Model Prior Prediction
Non-Word Recognition Times
Word Recognition Times
Frequency
Recognition times
0
20
40
60
80
100
0
100
200
300
400
500
600
700

4.5.4


Null Model / Observed (Tw)
Prior
Observed Data
Density
Recognition Time
Null Model / Observed (Tnw)
Prior
Observed Data
Density
Recognition Time
Lexical Model / Observed (Tw)
Prior
Observed Data
Density
Recognition Time
Lexical Model / Observed (Tnw)
Prior
Observed Data
Density
Recognition Time

```
4.5.5
```



In 4.5.3 In the Null Model, prior distributions of reaction times to words and non-words is extremely similar. However, in the Lexical-Access Model, there is, on average, a slightly longer reaction time for Non-Words as compared to Words. (The entire distribution seems to be shifted to the right).

In 4.5.4, The four histograms show that both models overlap considerably well with the observed data, and hence both work well. On comparing them, we observe that for Non word-Recognition times, Null hypothesis is closer to the given data as compared to the Lexical access,on comparing the peaks,whereas, for word-recognition times, both models were quite close to observed data.