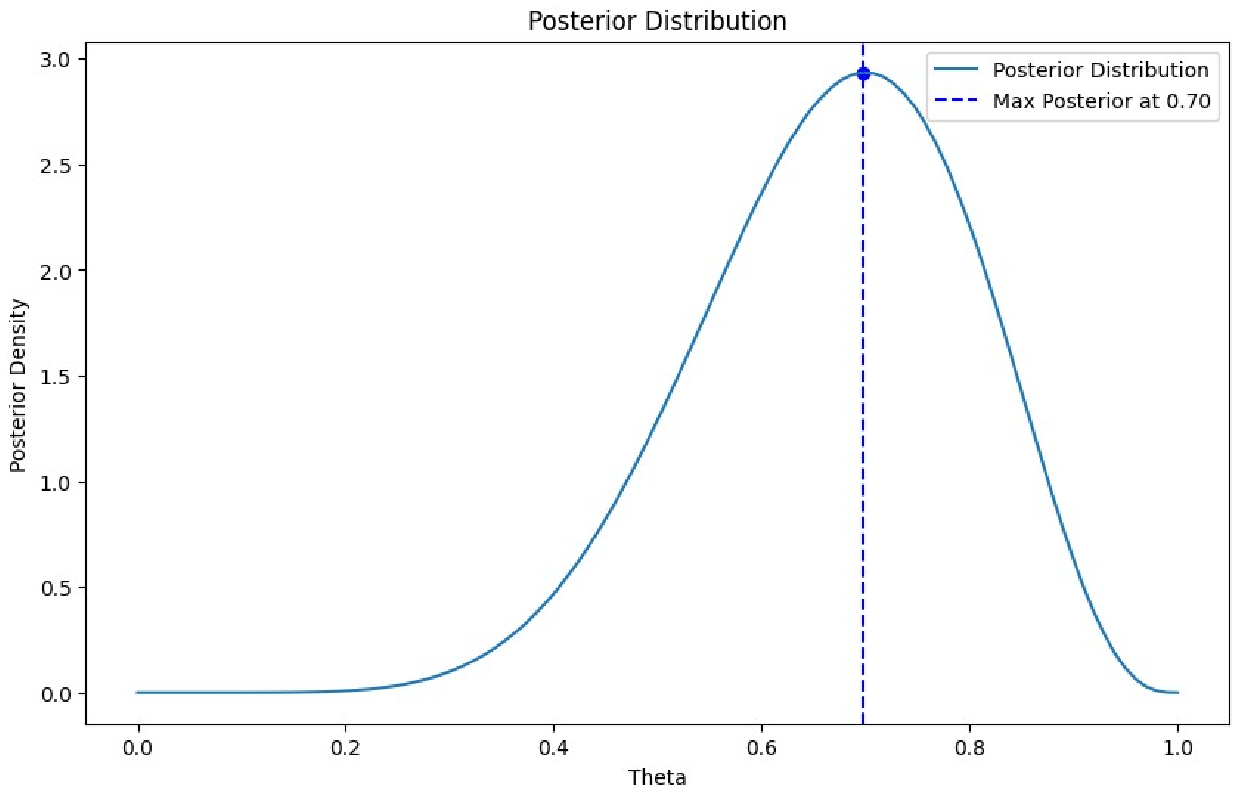
Part 1: A simple binomial model

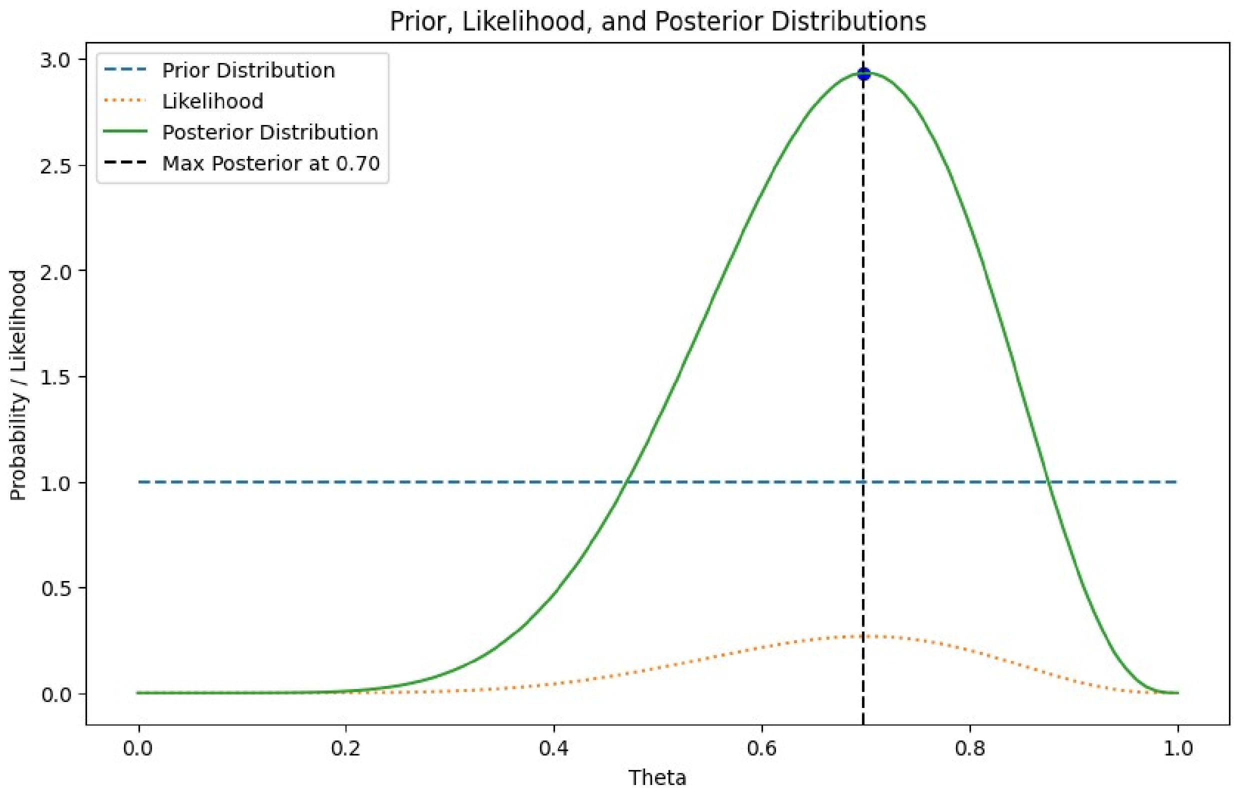
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import factorial

def calculate_likelihood(y, theta):
    return (factorial(10, exact=True) / (factorial(y, exact=True) *
factorial(10 - y, exact=True))) * (theta ** y) * ((1 - theta) ** (10 -
y))

def calculate_prior(theta):
    if 0 <= theta <= 1:
        return 1
    else:
        return 0

def calculate_posterior(y, theta):
    return calculate_likelihood(y, theta) * calculate_prior(theta) *
11

def display_results(theta_arr, y):
    print("1.1(a) Posterior Density for Different Values of Theta:\n")
    for theta in theta_arr:
        print(f'Theta = {theta : .3f} => Posterior Density =
{calculate_posterior(y, theta) : .6f}')
        print("\n")

    x_theta = np.linspace(0, 1, 100)
    y_posterior = [calculate_posterior(y, theta) for theta in x_theta]
    y_prior = np.array([calculate_prior(theta) for theta in x_theta])
    y_likelihood = np.array([calculate_likelihood(y, theta) for theta
in x_theta])

    max_index = np.argmax(y_posterior)
    max_theta = x_theta[max_index]
    max_posterior = y_posterior[max_index]

    plt.figure(figsize=(10, 6))
    plt.plot(x_theta, y_posterior, label='Posterior Distribution',
linestyle='-')
    plt.axvline(max_theta, color='b', linestyle='--', label=f'Max
Posterior at {max_theta:.2f}')
    plt.scatter(max_theta, max_posterior, color='b')
    plt.xlabel('Theta')
    plt.ylabel('Posterior Density')
    plt.title('Posterior Distribution')
    plt.legend()
    print("1.2 Posterior Distribution Graph:\n")
    plt.show()
    print("\n")
```

```
    plt.figure(figsize=(10, 6))
    plt.plot(x_theta, y_prior, label='Prior Distribution',
linestyle='--')
    plt.plot(x_theta, y_likelihood, label='Likelihood', linestyle=':')
    plt.plot(x_theta, y_posterior, label='Posterior Distribution')
    plt.axvline(max_theta, color='k', linestyle='--', label=f'Max
Posterior at {max_theta:.2f}')
    plt.scatter(max_theta, max_posterior, color='b')
    plt.xlabel('Theta')
    plt.ylabel('Probability / Likelihood')
    plt.title('Prior, Likelihood, and Posterior Distributions')
    plt.legend()
    print("1.3 Maximum Posterior Density:\n")
    print(f'Maximum posterior value is at theta = {max_theta:.2f} with
a value of {max_posterior:.6f}')
    print("\n")
    print("1.4 Comparison of Prior, Likelihood, and Posterior
Distributions:\n")
    plt.show()
    print("\n")

y = 7
theta_arr = [0.75, 0.25, 1]
display_results(theta_arr, y)
```

```
1.1(a) Posterior Density for Different Values of Theta:

Theta =  0.750 => Posterior Density =  2.753105


Theta =  0.250 => Posterior Density =  0.033989


Theta =  1.000 => Posterior Density =  0.000000


1.2 Posterior Distribution Graph:
```




```
1.3 Maximum Posterior Density:

Maximum posterior value is at theta = 0.70 with a value of 2.934468


1.4 Comparison of Prior, Likelihood, and Posterior Distributions:
```



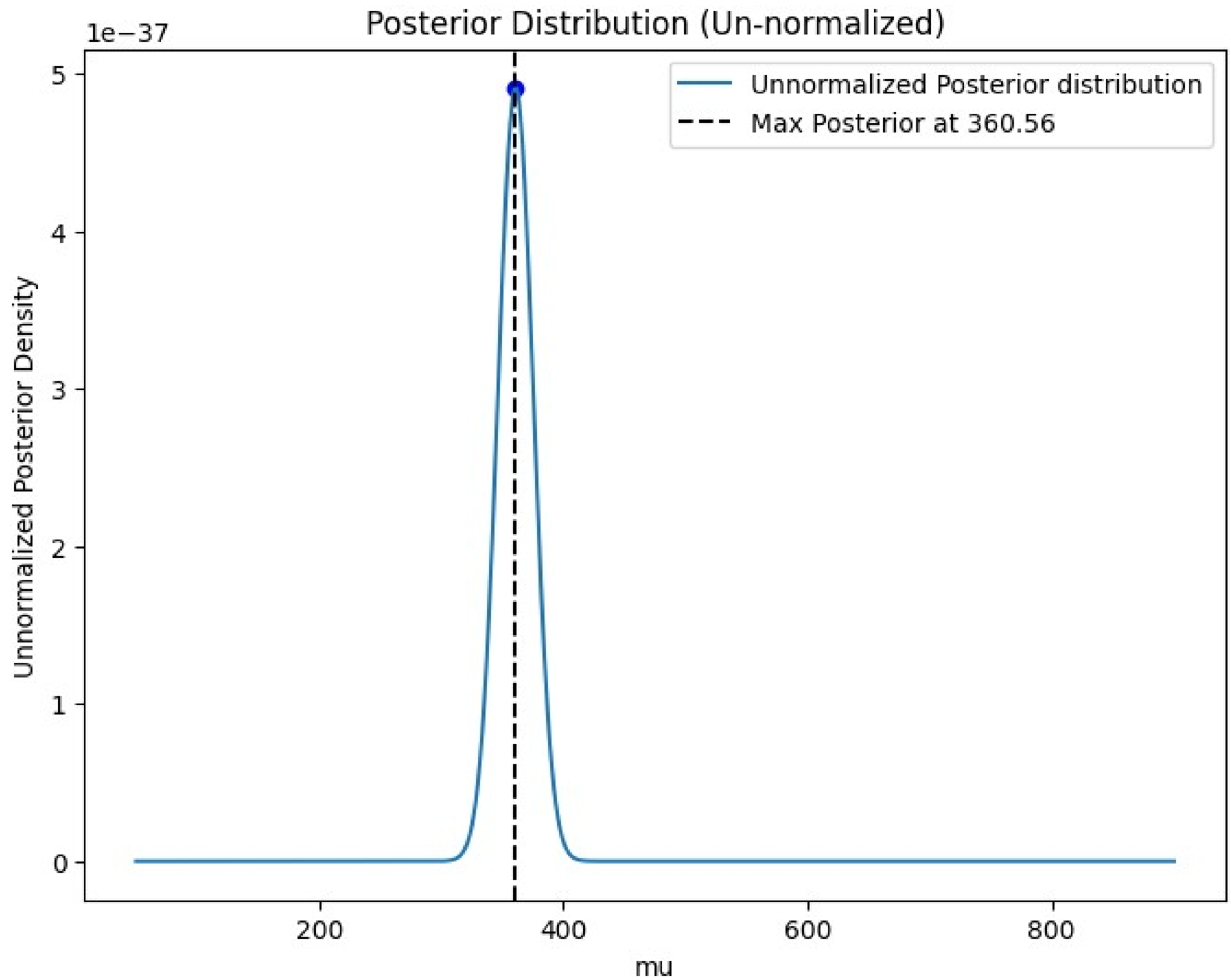
Prior, Likelihood, and Posterior Distributions
Prior Distribution
Likelihood
Posterior Distribution
Max Posterior at 0.70
Probability / Likelihood
Theta
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0.2
0.4
0.6
0.8

Part 2: A Gaussian model of reading

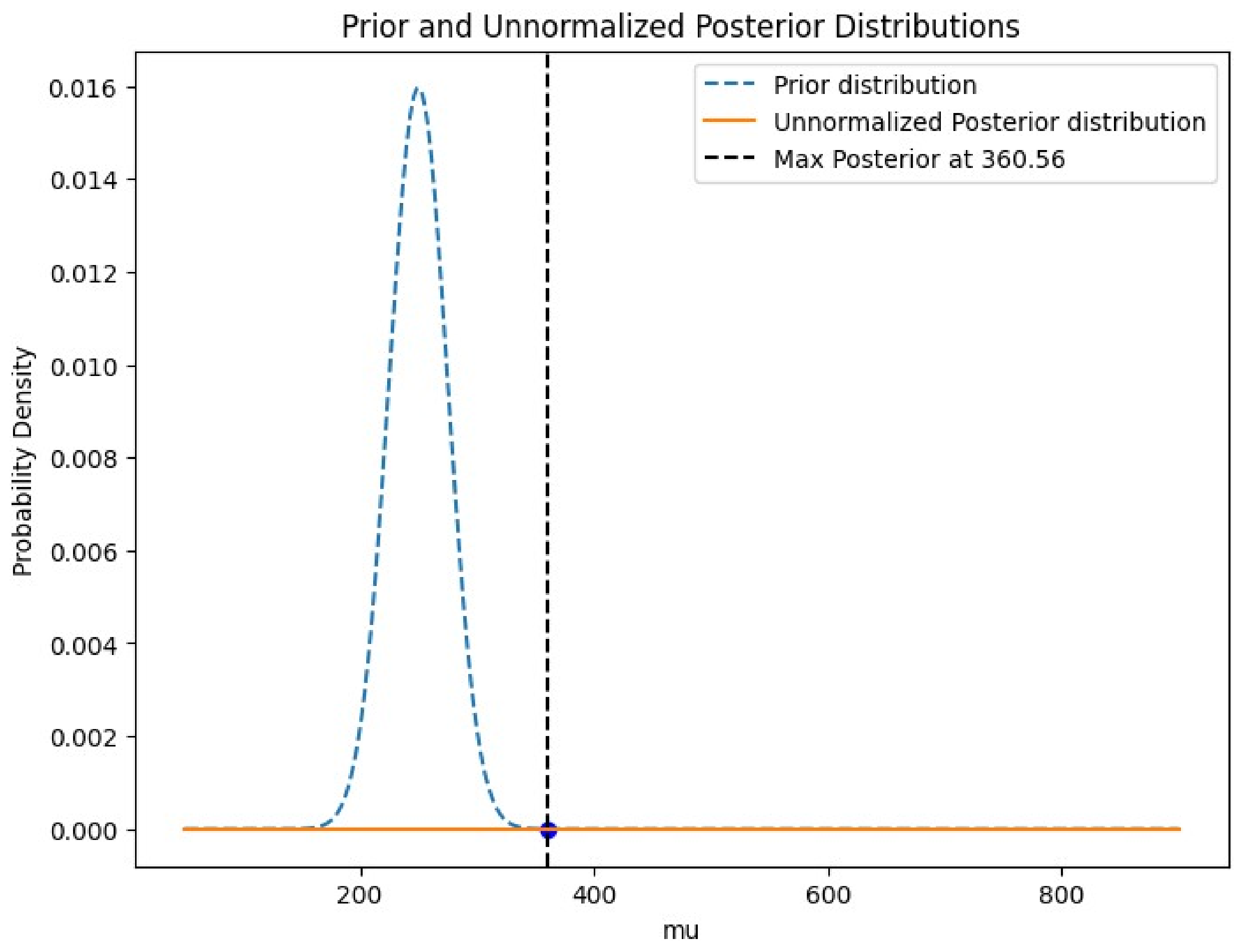
```
import numpy as np
import matplotlib.pyplot as plt

def calculate_likelihood(y, mu, sigma):
    n = len(y)
    likelihood_val = (1 / ((sigma * np.sqrt(2 * np.pi)) ** n)) *
np.exp(-(np.sum((y - mu) ** 2)) / (2 * (sigma ** 2)))
    return likelihood_val

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

def calculate_posterior(y, mu, sigma):
    return calculate_likelihood(y, mu, sigma) * calculate_prior(mu)

y = np.array([300.0, 270.0, 390.0, 450.0, 500.0, 290.0, 680.0, 450.0])
sigma = 50.0
mu_arr = [300.0, 900.0, 50.0]

print("\033[96m(a). Unnormalized Posterior Density:\033[0m\n")
for mu in mu_arr:
    print(f'\033[94mValue of Unnormalized Posterior at mean => {mu
: .2f} is => ', calculate_posterior(y, mu, sigma))
    print("\n")

x_mu = np.linspace(50, 900, 1000)
y_posterior = [calculate_posterior(y, mu, sigma) for mu in x_mu]
y_prior = np.array([calculate_prior(mu) for mu in x_mu])

max_index = np.argmax(y_posterior)
max_mu = x_mu[max_index]
max_posterior = y_posterior[max_index]

plt.figure(figsize=(8, 6))
plt.plot(x_mu, y_posterior, label='Unnormalized Posterior
distribution', linestyle='-')
plt.axvline(max_mu, color='k', linestyle='--', label=f'Max Posterior
at {max_mu:.2f}')
plt.scatter(max_mu, max_posterior, color='b')
plt.xlabel('mu')
plt.ylabel('Unnormalized Posterior Density')
plt.title('Posterior Distribution (Un-normalized)')
plt.legend()
print("\033[96m(b). Unnormalized Posterior Distribution Graph:\033[0m\
n")
plt.show()
print("\n")
```

```
plt.figure(figsize=(8, 6))
plt.plot(x_mu, y_prior, label='Prior distribution', linestyle='--')
plt.plot(x_mu, y_posterior, label='Unnormalized Posterior
distribution')
plt.axvline(max_mu, color='k', linestyle='--', label=f'Max Posterior
at {max_mu:.2f}')
plt.scatter(max_mu, max_posterior, color='b')
plt.xlabel('mu')
plt.ylabel('Probability Density')
plt.title('Prior and Unnormalized Posterior Distributions')
plt.legend()
print("\033[96m(c.1). Explanation:\033[0m\n")
plt.show()
print("\n The value of Unnormalized Posterior Distribution,it appears
to be a straight line so changing scale\n")

y_posterior_upscaled = [(calculate_posterior(y, mu, sigma)) * 1e+34
for mu in x_mu] # Up-scaled the value of Posterior-Distribution.

plt.figure(figsize=(8, 6))
plt.plot(x_mu, y_prior, label='Prior distribution', linestyle='--')
plt.plot(x_mu, y_posterior_upscaled, label='Unnormalized (scaled)
Posterior distribution (x 1e+34)')
plt.axvline(max_mu, color='k', linestyle='--', label=f'Max Posterior
at {max_mu:.2f}')
plt.scatter(max_mu, max_posterior, color='b')
plt.xlabel('mu')
plt.ylabel('Probability Density')
plt.title('Prior and Unnormalized (scaled) Posterior Distributions')
plt.legend()
print("\033[96m(c.2). Scaled Unnormalized Posterior Distribution
Graph:\033[0m\n")
plt.show()
```

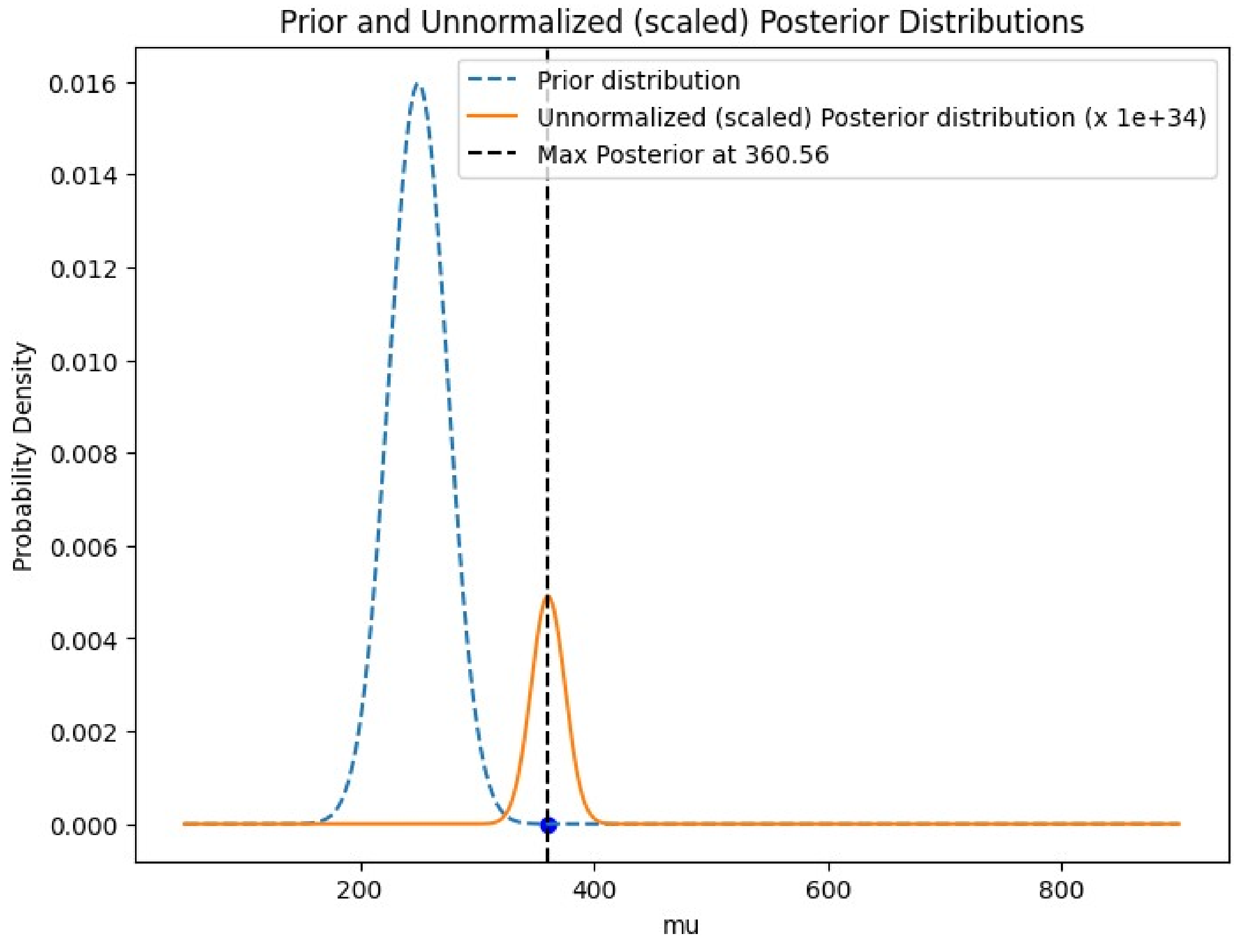
```
(a). Unnormalized Posterior Density:

Value of Unnormalized Posterior at mean =>  300.00 is =>
6.824247957486409e-41


Value of Unnormalized Posterior at mean =>  900.00 is =>  0.0


Value of Unnormalized Posterior at mean =>  50.00 is =>
9.691373559300655e-138


(b). Unnormalized Posterior Distribution Graph:
```




```
(c.1). Explanation:
```




```
 The value of Unnormalized Posterior Distribution,it appears to be a
straight line so changing scale

(c.2). Scaled Unnormalized Posterior Distribution Graph:
```

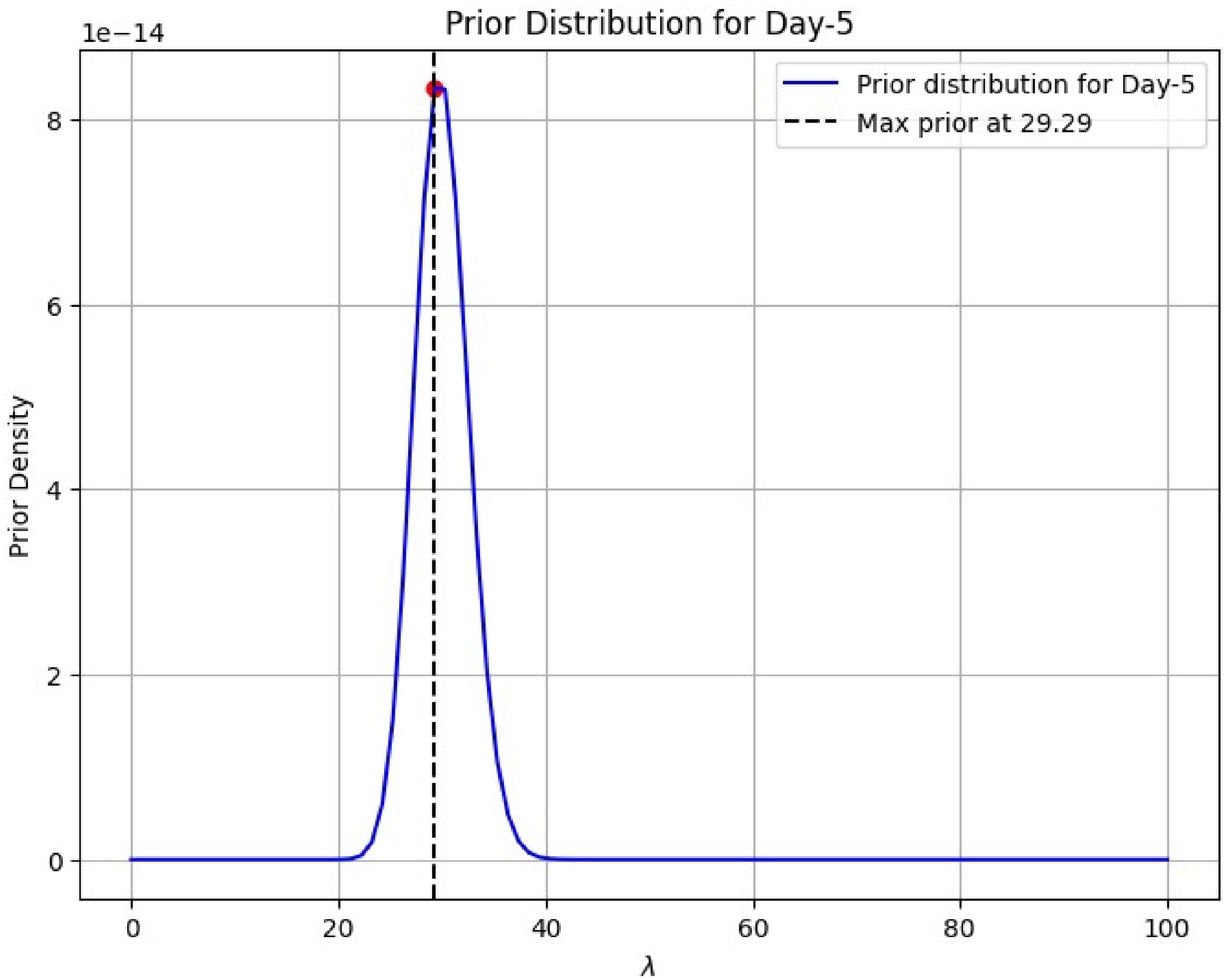

Prior and Unnormalized (scaled) Posterior Distributions
Prior distribution
Unnormalized (scaled) Posterior distribution (x 1e+34)
Max Posterior at 360.56
Probability Density
0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
200
400
600
800
mu

Part 3: The Bayesian learning

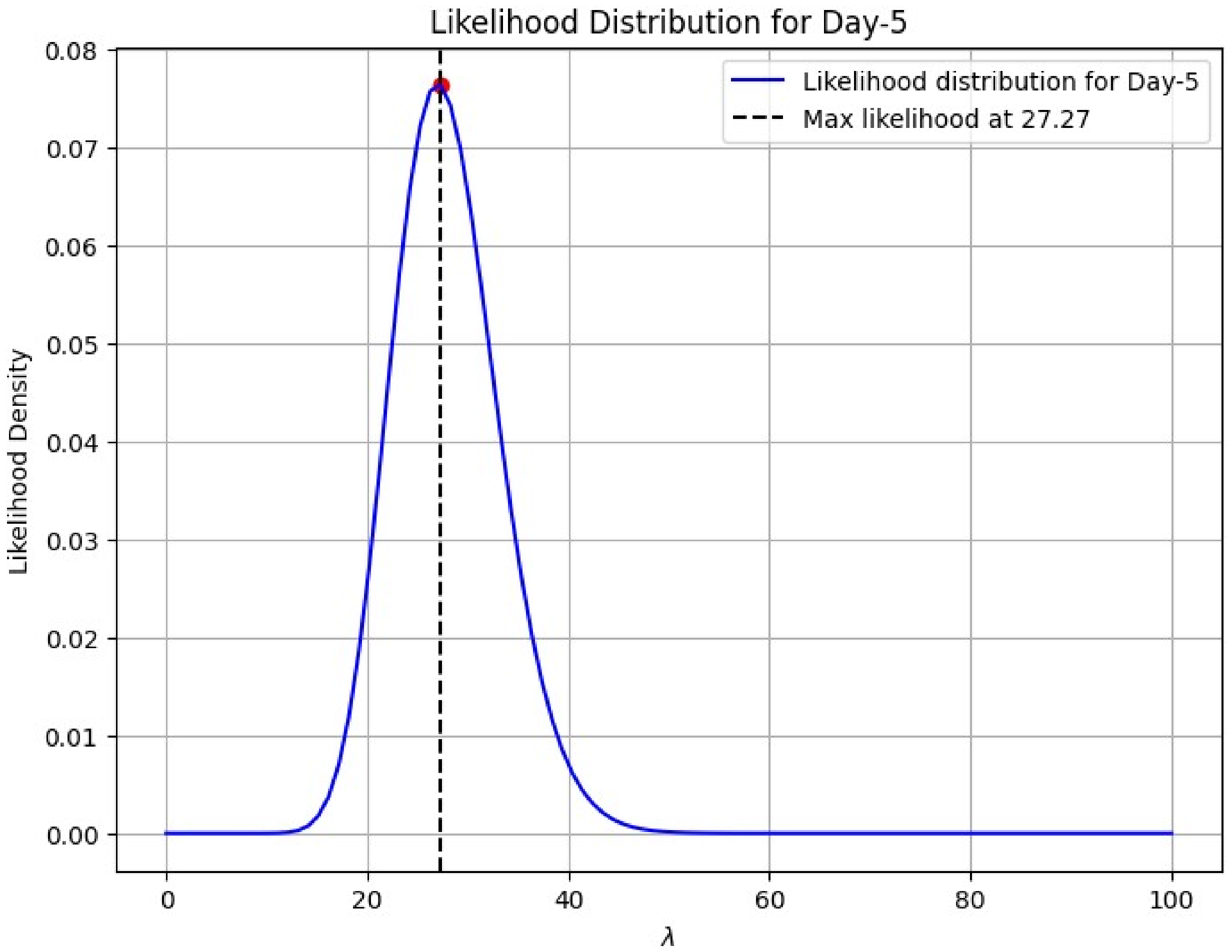
```
from scipy.stats import gamma
import numpy as np
import matplotlib.pyplot as plt

def likelihood_custom(lamda, k):
    return (lamda ** k) * np.exp(-lamda) / np.math.factorial(int(k))

def prior_custom(lamda):
    return gamma.pdf(lamda, a=40, scale=2)

accidents = [25, 20, 23, 27]
lamda = np.linspace(0, 100, 100)
prior_ = np.zeros(100)
likelihood_ = np.zeros(100)
posterior_ = np.zeros(100)

for i in range(0, 100):
    prior_[i] = prior_custom(lamda[i])

for k in accidents:
    for i in range(0, 100):
        likelihood_[i] = likelihood_custom(lamda[i], k)
        posterior_[i] = likelihood_[i] * prior_[i]
    # Update prior to posterior for current day
    prior_ = posterior_

max_index = np.argmax(prior_)
max_lamda = lamda[max_index]
max_prior = prior_[max_index]

plt.figure(figsize=(8, 6))
plt.grid()
plt.plot(lamda, prior_, label='Prior distribution for Day-5',
linestyle='-', color='blue')
plt.axvline(max_lamda, color='black', linestyle='--', label=f'Max
prior at {max_lamda:.2f}')
plt.scatter(max_lamda, max_prior, color='red')
plt.xlabel('$\lambda$')
plt.ylabel('Prior Density')
plt.title('Prior Distribution for Day-5')
plt.legend()
print("Analytically, the Prior-Distribution for Day - 5 will be:
Gamma(135.0, 6.0)\n")
plt.show()

max_index = np.argmax(likelihood_)
max_lamda = lamda[max_index]
max_likelihood = likelihood_[max_index]

plt.figure(figsize=(8, 6))
```

```python
plt.grid()
plt.plot(lamda, likelihood_, label='Likelihood distribution for Day-
5', linestyle='-', color='blue')
plt.axvline(max_lamda, color='black', linestyle='--', label=f'Max
likelihood at {max_lamda:.2f}')
plt.scatter(max_lamda, max_likelihood, color='red')
plt.xlabel('$\lambda$')
plt.ylabel('Likelihood Density')
plt.title('Likelihood Distribution for Day-5')
plt.legend()
plt.show()

accidents_assumption = np.linspace(0, 100, 50)
lambda_new = np.linspace(10, 50, 50)
prediction_acc = []

plt.figure(figsize=(8, 6))
for lamda in lambda_new:
    likelihood_pred = [likelihood_custom(lamda, int(k)) for k in
accidents_assumption]
    prediction_acc.append(likelihood_pred)

prediction_acc = np.array(prediction_acc)
mean_prediction = np.mean(prediction_acc, axis=0)

plt.plot(accidents_assumption, mean_prediction, label='Likelihood
distribution for Day-5', linestyle='-', color='blue')
max_index = np.argmax(mean_prediction)
max_k = accidents_assumption[max_index]
max_lik = mean_prediction[max_index]

plt.axvline(max_k, color='black', linestyle='--', label=f'Max
likelihood at {max_k:.2f}')
plt.scatter(max_k, max_lik, color='red')
plt.xlabel('k')
plt.ylabel('Likelihood Density')
plt.title('Accident Prediction for Day-5')
plt.legend()
plt.show()

print(f'Predicted Number of accidents on Day - 5 will be around 24:')
```

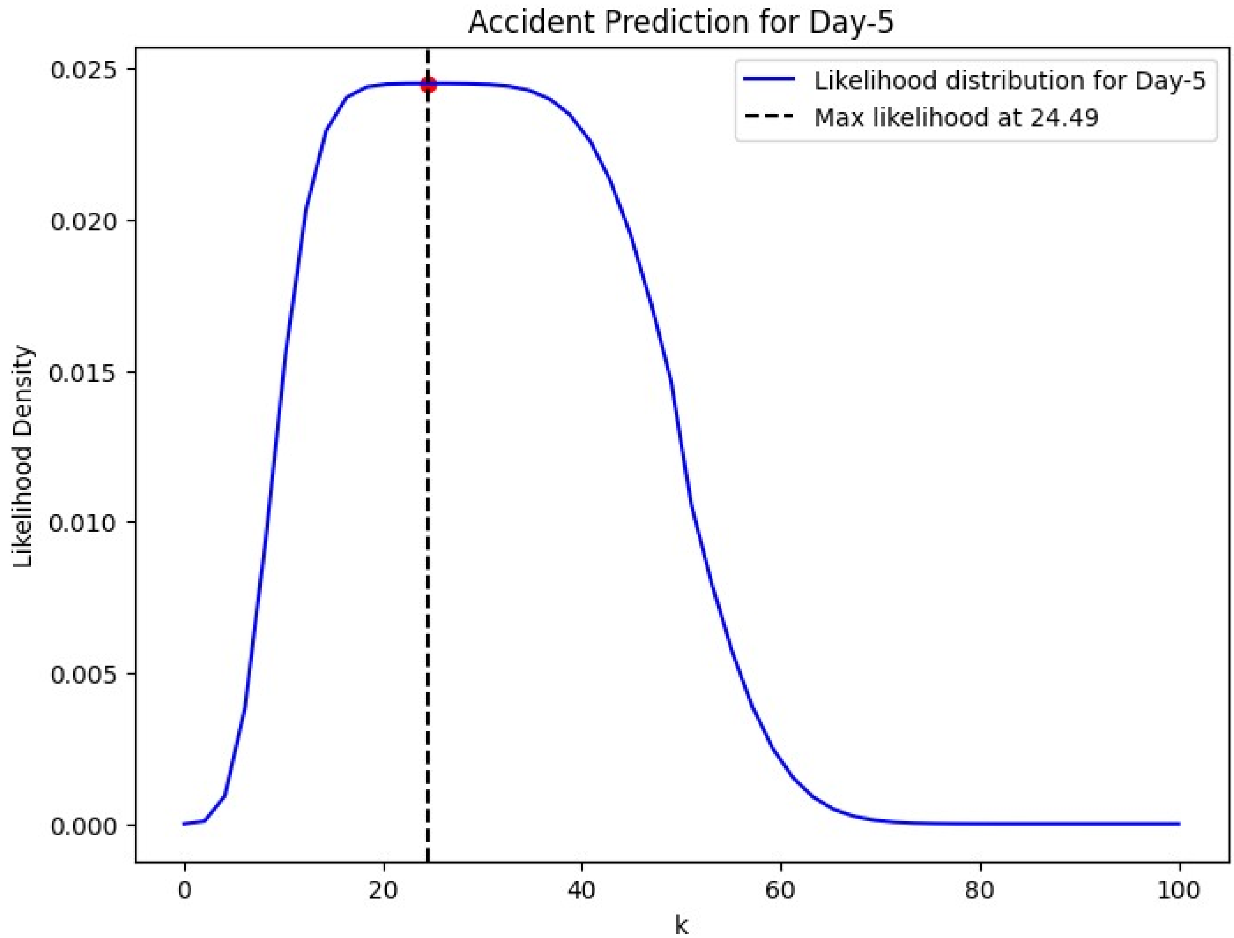
```
<ipython-input-20-0b063a8fda58>:6: DeprecationWarning: `np.math` is a
deprecated alias for the standard library `math` module (Deprecated
Numpy 1.25). Replace usages of `np.math` with `math`
  return (lamda ** k) * np.exp(-lamda) / np.math.factorial(int(k))

Analytically, the Prior-Distribution for Day - 5 will be: Gamma(135.0,
6.0)
```



Prior Distribution for Day-5
1e−14
Prior distribution for Day-5
Max prior at 29.29
Prior Density
λ
0
2
4
6
8
0
20
40
60
80
100



Likelihood Distribution for Day-5
Likelihood distribution for Day-5
Max likelihood at 27.27
Likelihood Density
λ
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
20
40
60
80
100




```
Predicted Number of accidents on Day - 5 will be around 24:
```

Part 4: Model building in the Bayesian framework

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from scipy.stats import norm

url =
"https://raw.githubusercontent.com/yadavhimanshu059/CGS698C/main/notes
/Module-2/recognition.csv"
dat = pd.read_csv(url)
dat = dat.iloc[:, 1:]

def likelihood_estimation(mu, sigma, delta):
    return ((np.prod(norm.pdf(dat["Tw"], mu, sigma))) *
(np.prod(norm.pdf(dat["Tnw"], mu + delta, sigma))))

# Unnormalized Posterior Prediction for Null Hypothesis.
x_label = np.linspace(200, 400, 200)
sigma = 60.0
y_posterior_null = [(norm.pdf(x, 300, 50)*likelihood_estimation(x,
sigma, 0.0)) for x in x_label]
plt.figure(figsize=(8, 6))
plt.plot(x_label, y_posterior_null, label='Null Hypothesis',
linestyle='-')
max_index = np.argmax(y_posterior_null)
max_mu = x_label[max_index]
max_posterior = y_posterior_null[max_index]
plt.axvline(max_mu, color='k', linestyle='--', label=f'Max Posterior
at {max_mu:.2f}')
plt.scatter(max_mu, max_posterior, color='b')
plt.xlabel('Word Recognition Time')
plt.ylabel('Unnormalized Posterior Density')
plt.title('Unnormalized Posterior Distribution (Null Hypothesis)')
plt.legend()
print("(a).\n")
plt.show()

# Prior Predictions for Lexical-access model.
num_samples = 100000
mu_samples = np.random.normal(300, 50, num_samples)
delta_samples = np.abs(np.random.normal(0, 50, num_samples))
word_recognition_times = np.random.normal(mu_samples, sigma)
nonword_recognition_times = np.random.normal(mu_samples +
delta_samples, sigma)
plt.figure(figsize=(10, 5))
plt.hist(word_recognition_times, bins=50, alpha=0.75,edgecolor='k',
label='Word Recognition Times')
plt.xlabel('Word Recognition Time')
plt.ylabel('Frequency')
plt.title('Histograms of Word Recognition Times')
```

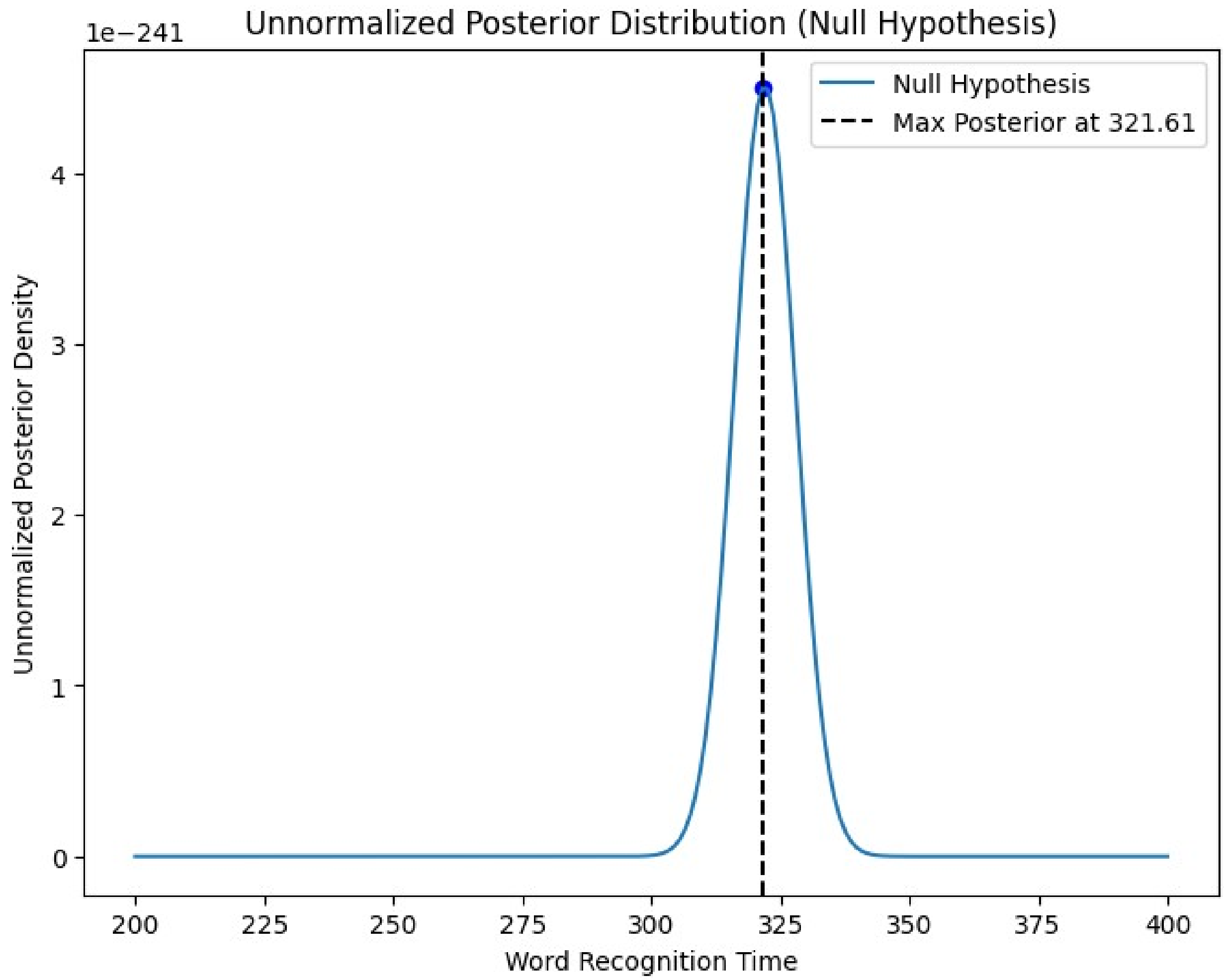
```
plt.legend()
print("\n\n(b).\n")
plt.show()

plt.figure(figsize=(10, 5))
plt.hist(nonword_recognition_times, bins=50, alpha=0.75,edgecolor='k',
label='Non-word Recognition Times')
plt.xlabel('Non-word Recognition Time')
plt.ylabel('Frequency')
plt.title('Histograms of Non-word Recognition Times')
plt.legend()
plt.show()

# Comparison between Prior Prediction of both the models.
print("\n\n(c).\n")
print("Null-Hypothesis model (delta = 0) : It will follow the
histogram of 'Word Recognition Times' for both 'Tw' and 'Tnw'.")
print("Hence, for 'Word Recognition Times' both the models use the
same prior.")
print("For 'Non-word Recognition Times' - \n")

mean_word = np.mean(word_recognition_times)
median_word = np.median(word_recognition_times)
mean_nonword = np.mean(nonword_recognition_times)
median_nonword = np.median(nonword_recognition_times)

fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10, 5))

# Plot non-word recognition times histogram for NULL HYPOTHESIS.
counts, bins, patches = axes[0].hist(word_recognition_times, bins=50,
alpha=0.5, label='Non-Word Recognition Times', edgecolor='black')
bin_centers = 0.5 * (bins[:-1] + bins[1:])
axes[0].plot(bin_centers, counts, linestyle='-', marker='.',
color='k')
axes[0].axvline(mean_word, color='blue', linestyle='dashed',
linewidth=1, label=f'Mean: {mean_word:.2f}')
axes[0].axvline(median_word, color='green', linestyle='dashed',
linewidth=1, label=f'Median: {median_word:.2f}')
axes[0].set_xlabel('Non-Word Recognition Time')
axes[0].set_ylabel('Frequency')
axes[0].set_title('NULL HYPOTHESIS')
axes[0].legend()
counts, bins, patches = axes[1].hist(nonword_recognition_times,
bins=50, alpha=0.5, label='Non-word Recognition Times',
edgecolor='black')
bin_centers = 0.5 * (bins[:-1] + bins[1:])
axes[1].plot(bin_centers, counts, linestyle='-', marker='.',
color='k')
axes[1].axvline(mean_nonword, color='blue', linestyle='dashed',
linewidth=1, label=f'Mean: {mean_nonword:.2f}')
```

```
axes[1].axvline(median_nonword, color='green', linestyle='dashed',
linewidth=1, label=f'Median: {median_nonword:.2f}')
axes[1].set_xlabel('Non-word Recognition Time')
axes[1].set_ylabel('Frequency')
axes[1].set_title('LEXICAL-ACCESS HYPOTHESIS')
axes[1].legend()
plt.tight_layout()
plt.show()
print("\n")
print(f'NULL HYPOTHESIS Prior: Mean = {mean_word:.2f}, Median =
{median_word:.2f}')
print(f'LEXICAL-ACCESS HYPOTHESIS Prior: Mean = {mean_nonword:.2f},
Median = {median_nonword:.2f}')
print("=> Predicting a non-word in Lexical Hypothesis takes more time
than that of Null Hypothesis.\n")
# Comparing Prior Predictions against the given data.
# For NULL HYPOTHESIS :
print("(d).\n")
fig, axes = plt.subplots(nrows=4, ncols=2, figsize=(14, 10))
counts, bins, patches = axes[0, 0].hist(word_recognition_times,
bins=50, alpha=0.5, label='Word Recognition Times', edgecolor='black')
axes[0, 0].axvline(mean_word, color='blue', linestyle='dashed',
linewidth=1, label=f'Mean: {mean_word:.2f}')
axes[0, 0].axvline(median_word, color='green', linestyle='dashed',
linewidth=1, label=f'Median: {median_word:.2f}')
axes[0, 0].set_xlabel('Word Recognition Time')
axes[0, 0].set_ylabel('Frequency')
axes[0, 0].set_title('NULL HYPOTHESIS')
axes[0, 0].legend()
counts, bins, patches = axes[0, 1].hist(dat["Tw"], bins=50, alpha=0.5,
label='Given Data', edgecolor='black')
axes[0, 1].axvline(np.mean(dat["Tw"]), color='blue',
linestyle='dashed', linewidth=1, label=f'Mean:
{np.mean(dat["Tw"]):.2f}')
axes[0, 1].axvline(np.median(dat["Tw"]), color='green',
linestyle='dashed', linewidth=1, label=f'Median:
{np.median(dat["Tw"]):.2f}')
axes[0, 1].set_xlabel('Word Recognition Time')
axes[0, 1].set_ylabel('Frequency')
axes[0, 1].set_title('Given Data')
axes[0, 1].legend()
counts, bins, patches = axes[1, 0].hist(word_recognition_times,
bins=50, alpha=0.5, label='Non-Word Recognition Times',
edgecolor='black')
axes[1, 0].axvline(mean_word, color='blue', linestyle='dashed',
linewidth=1, label=f'Mean: {mean_word:.2f}')
axes[1, 0].axvline(median_word, color='green', linestyle='dashed',
linewidth=1, label=f'Median: {median_word:.2f}')
axes[1, 0].set_xlabel('Non-Word Recognition Time')
```

```
axes[1, 0].set_ylabel('Frequency')
axes[1, 0].set_title('NULL HYPOTHESIS')
axes[1, 0].legend()
counts, bins, patches = axes[1, 1].hist(dat["Tnw"], bins=50,
alpha=0.5, label='Given Data', edgecolor='black')
axes[1, 1].axvline(np.mean(dat["Tnw"]), color='blue',
linestyle='dashed', linewidth=1, label=f'Mean:
{np.mean(dat["Tnw"]):.2f}')
axes[1, 1].axvline(np.median(dat["Tnw"]), color='green',
linestyle='dashed', linewidth=1, label=f'Median:
{np.median(dat["Tnw"]):.2f}')
axes[1, 1].set_xlabel('Non-Word Recognition Time')
axes[1, 1].set_ylabel('Frequency')
axes[1, 1].set_title('Given Data')
axes[1, 1].legend()

# For Lexical-Access Hypothesis.
counts, bins, patches = axes[2, 0].hist(word_recognition_times,
bins=50, alpha=0.5, label='Word Recognition Times', edgecolor='black')
axes[2, 0].axvline(mean_word, color='blue', linestyle='dashed',
linewidth=1, label=f'Mean: {mean_word:.2f}')
axes[2, 0].axvline(median_word, color='green', linestyle='dashed',
linewidth=1, label=f'Median: {median_word:.2f}')
axes[2, 0].set_xlabel('Word Recognition Time')
axes[2, 0].set_ylabel('Frequency')
axes[2, 0].set_title('LEXICAL-ACCESS HYPOTHESIS')
axes[2, 0].legend()
counts, bins, patches = axes[2, 1].hist(dat["Tw"], bins=50, alpha=0.5,
label='Given Data', edgecolor='black')
axes[2, 1].axvline(np.mean(dat["Tw"]), color='blue',
linestyle='dashed', linewidth=1, label=f'Mean:
{np.mean(dat["Tw"]):.2f}')
axes[2, 1].axvline(np.median(dat["Tw"]), color='green',
linestyle='dashed', linewidth=1, label=f'Median:
{np.median(dat["Tw"]):.2f}')
axes[2, 1].set_xlabel('Word Recognition Time')
axes[2, 1].set_ylabel('Frequency')
axes[2, 1].set_title('Given Data')
axes[2, 1].legend()
counts, bins, patches = axes[3, 0].hist(nonword_recognition_times,
bins=50, alpha=0.5, label='Non-Word Recognition Times',
edgecolor='black')
axes[3, 0].axvline(mean_nonword, color='blue', linestyle='dashed',
linewidth=1, label=f'Mean: {mean_nonword:.2f}')
axes[3, 0].axvline(median_nonword, color='green', linestyle='dashed',
linewidth=1, label=f'Median: {median_nonword:.2f}')
axes[3, 0].set_xlabel('Non-Word Recognition Time')
axes[3, 0].set_ylabel('Frequency')
axes[3, 0].set_title('LEXICAL-ACCESS HYPOTHESIS')
```

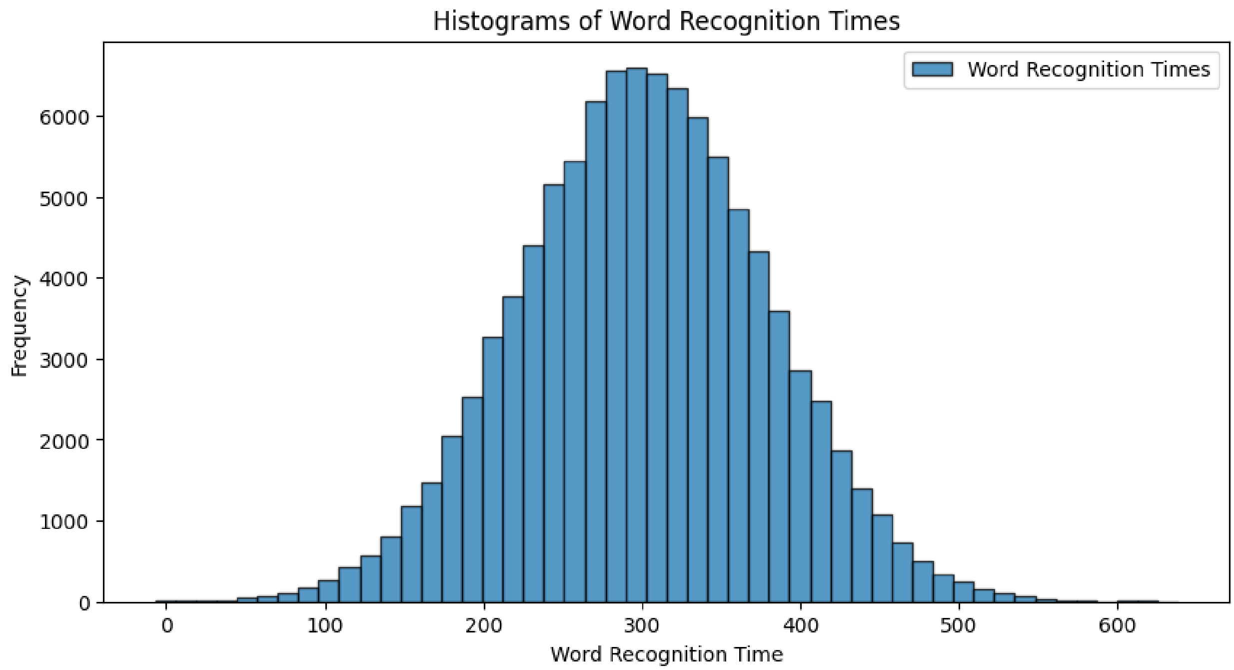
```
axes[3, 0].legend()
counts, bins, patches = axes[3, 1].hist(dat["Tnw"], bins=50,
alpha=0.5, label='Given Data', edgecolor='black')
axes[3, 1].axvline(np.mean(dat["Tnw"]), color='blue',
linestyle='dashed', linewidth=1, label=f'Mean:
{np.mean(dat["Tnw"]):.2f}')
axes[3, 1].axvline(np.median(dat["Tnw"]), color='green',
linestyle='dashed', linewidth=1, label=f'Median:
{np.median(dat["Tnw"]):.2f}')
axes[3, 1].set_xlabel('Non-Word Recognition Time')
axes[3, 1].set_ylabel('Frequency')
axes[3, 1].set_title('Given Data')
axes[3, 1].legend()
plt.tight_layout()
plt.show()
print("\nFor Null Hypothesis : \n")
print("Absolute Error in Prior Model for Mean Word-Recognition : ",
(np.abs(mean_word - np.mean(dat["Tw"]))/np.mean(dat["Tw"]))*100 )
print("Absolute Error in Prior Model for Mean Non-Word-Recognition :
", (np.abs(mean_word - np.mean(dat["Tnw"]))/np.mean(dat["Tnw"]))*100 )
print("\n")
print("For Lexical Hypothesis : \n")
print("Absolute Error in Prior Model for Mean Word-Recognition : ",
(np.abs(mean_word - np.mean(dat["Tw"]))/np.mean(dat["Tw"]))*100 )
print("Absolute Error in Prior Model for Mean Non-Word-Recognition :
", (np.abs(mean_nonword -
np.mean(dat["Tnw"]))/np.mean(dat["Tnw"]))*100 )
print("\n")
print("Since, Absolute Error for 'LEXICAL-ACCESS HYPOTHESIS' is less,
so it fits better.\n")
```

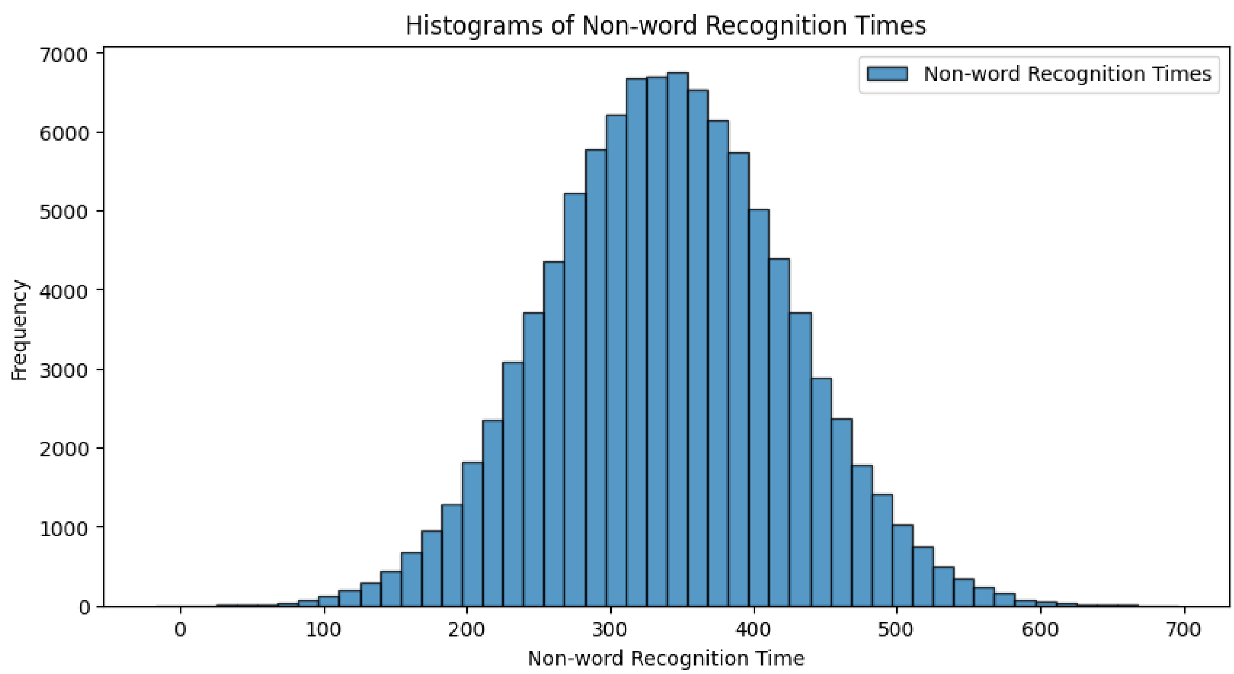
(a).



Unnormalized Posterior Distribution (Null Hypothesis)
1e−241
Null Hypothesis
Max Posterior at 321.61
Unnormalized Posterior Density
Word Recognition Time
0
1
2
3
4
200
225
250
275
300
325
350
375
400

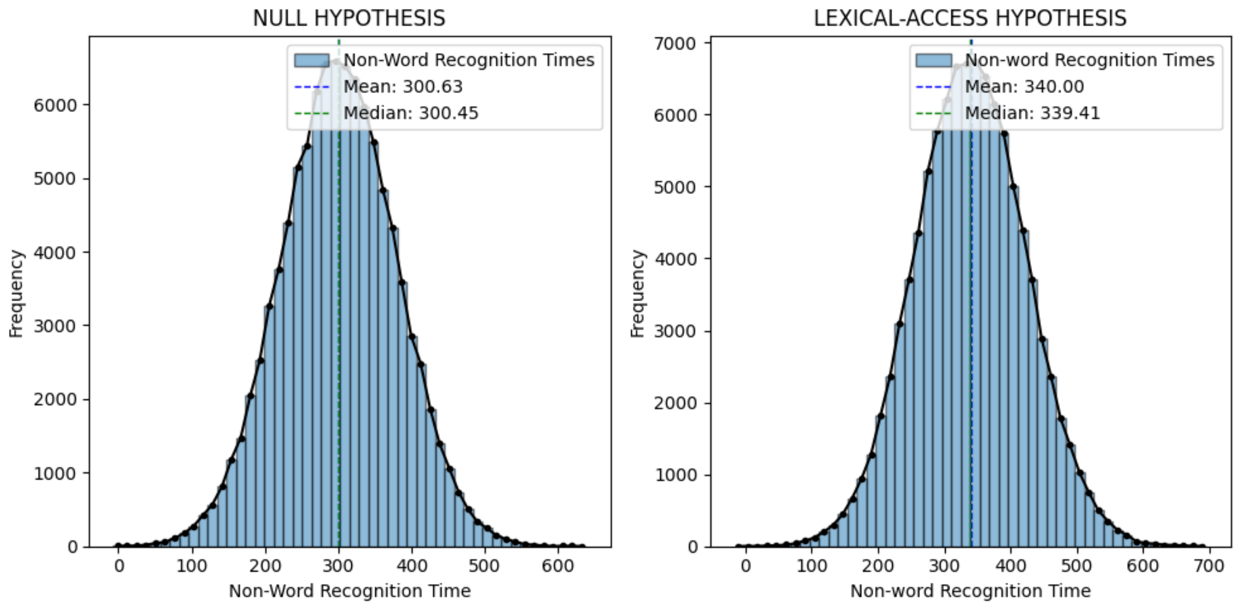
`(b).`



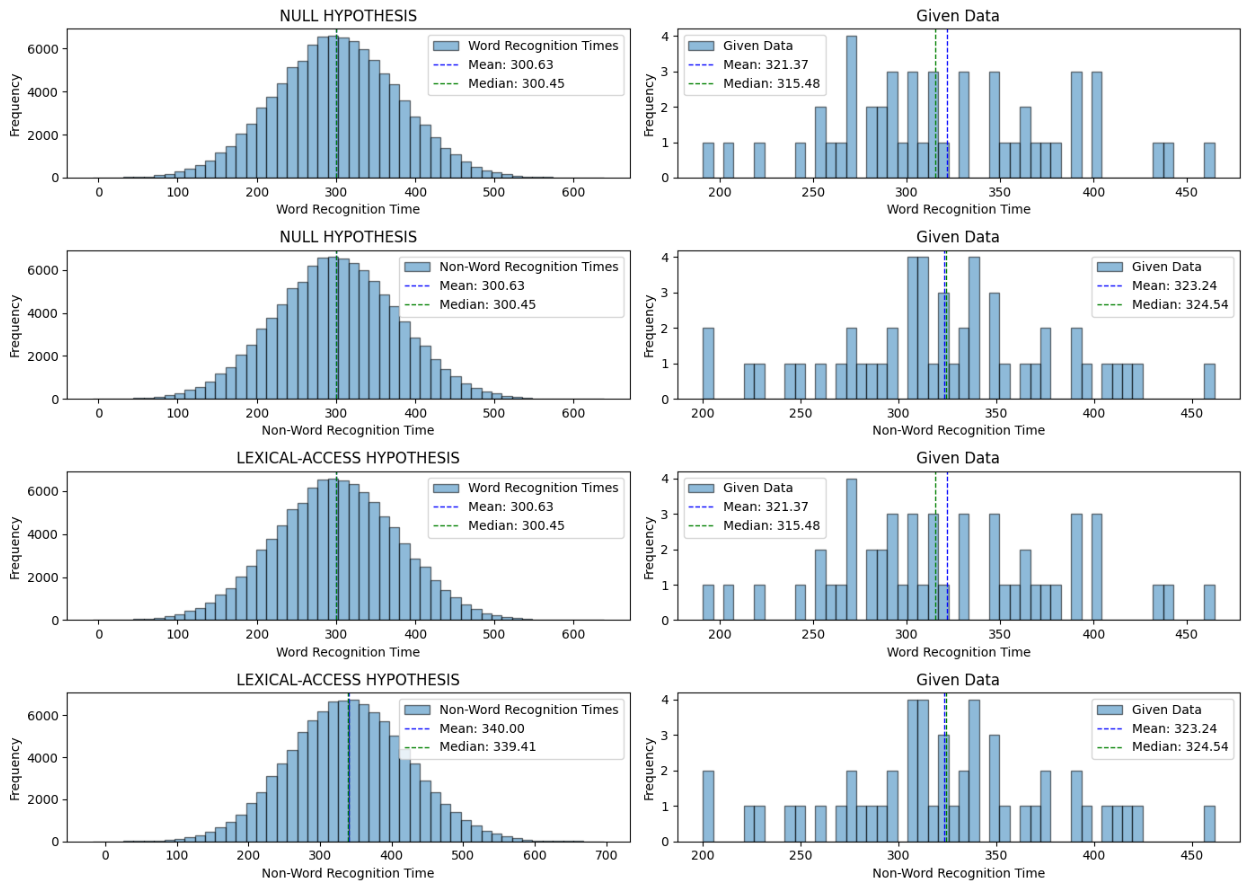



```
(c).

Null-Hypothesis model (delta = 0) : It will follow the histogram of
'Word Recognition Times' for both 'Tw' and 'Tnw'.
Hence, for 'Word Recognition Times' both the models use the same
prior.
```

```
For 'Non-word Recognition Times' -
```



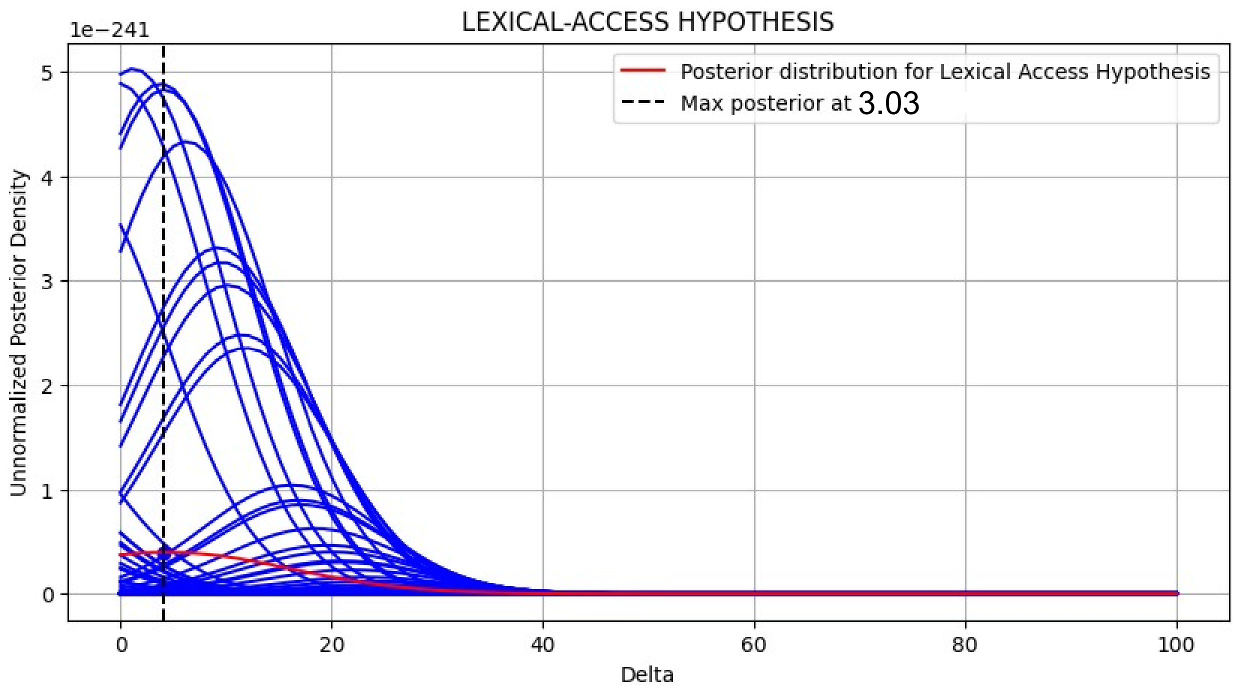

```
NULL HYPOTHESIS Prior: Mean = 300.63, Median = 300.45
LEXICAL-ACCESS HYPOTHESIS Prior: Mean = 340.00, Median = 339.41
=> Predicting a non-word in Lexical Hypothesis takes more time than
that of Null Hypothesis.

(d).
```




```
For Null Hypothesis :

Absolute Error in Prior Model for Mean Word-Recognition :
6.454470860822023
Absolute Error in Prior Model for Mean Non-Word-Recognition :
6.993981873266006


For Lexical Hypothesis :

Absolute Error in Prior Model for Mean Word-Recognition :
6.454470860822023
Absolute Error in Prior Model for Mean Non-Word-Recognition :
5.18407439652374


Since, Absolute Error for 'LEXICAL-ACCESS HYPOTHESIS' is less, so it
fits better.
```

```
print("(e).\n")
d_label = np.linspace(0, 100, 100)
```

```
mu_samples_ = np.random.normal(300, 50, 100)
posterior_distribution = []
fig, axes = plt.subplots(nrows=1, ncols=1, figsize=(10, 5))

for mu_ in mu_samples_:
    y_posterior_lex = [likelihood_estimation(mu_, sigma, delta) *
np.abs(norm.pdf(delta, 0, 50)) for delta in d_label]
    plt.plot(d_label, y_posterior_lex, label=None, linestyle='-',
color='b')
    posterior_distribution.append(y_posterior_lex)

posterior_distribution = np.array(posterior_distribution)
mean_posterior = np.mean(posterior_distribution, axis=0)
max_index = np.argmax(mean_posterior)
max_delta = d_label[max_index]
max_posterior = mean_posterior[max_index]

plt.plot(d_label, mean_posterior, label='Posterior distribution for
Lexical Access Hypothesis', linestyle='-', color='r')
plt.axvline(max_delta, color='k', linestyle='--', label=f'Max
posterior at {max_delta:.2f}')
plt.scatter(max_delta, max_posterior, color='b')
plt.xlabel('Delta')
plt.grid()
plt.ylabel('Unnormalized Posterior Density')
plt.title('LEXICAL-ACCESS HYPOTHESIS')
plt.legend()
plt.show()

(e).
```


LEXICAL-ACCESS HYPOTHESIS
1e−241
Posterior distribution for Lexical Access Hypothesis
Max posterior at 3.03
Unnormalized Posterior Density
Delta
0
1
2
3
4
5
0
20
40
60
80
100