A/B testing

BAYESIAN DATA ANALYSIS IN PYTHON

Michal Oleszak
Machine Learning Engineer

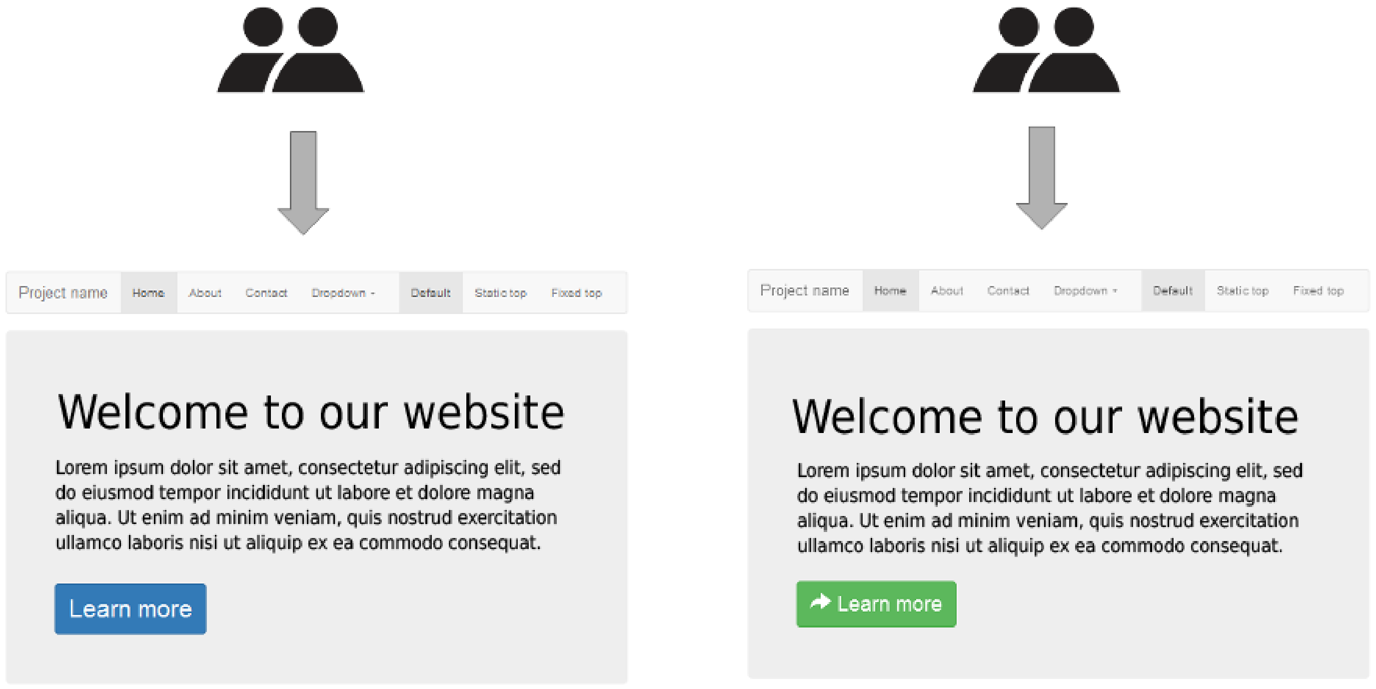
A/B testing

- Randomized experiment: divide users in two groups (A and B)

[1] Picture: adapted from https://commons.wikimedia.org/wiki/File:A-B_testing_simple_example.png

A/B testing

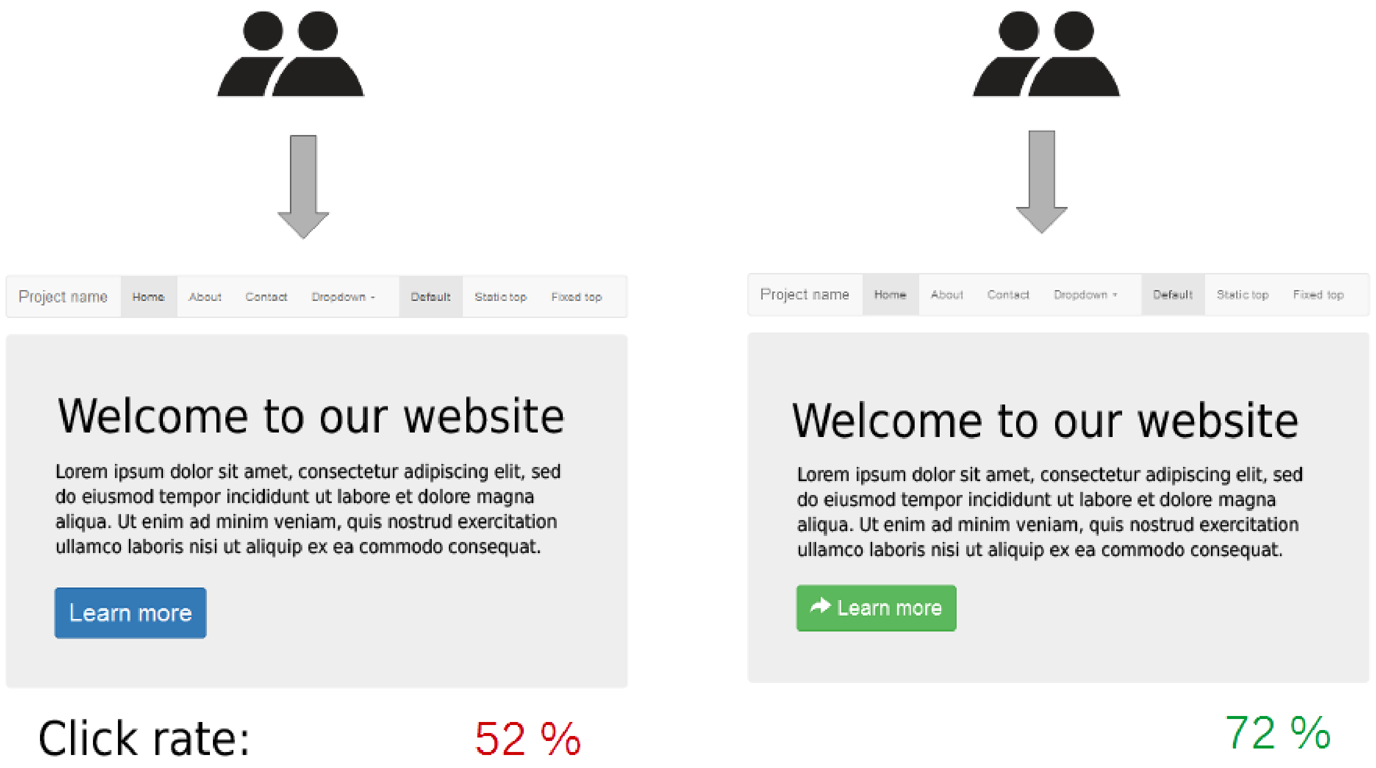
- Randomized experiment: divide users in two groups (A and B)
- Expose each group to a different version of something (e.g. website layout)



[1] Picture: adapted from https://commons.wikimedia.org/wiki/File:A-B_testing_simple_example.png

A/B testing

- Randomized experiment: divide users in two groups (A and B)
- Expose each group to a different version of something (e.g. website layout)
- Compare which group scores better on some metric (e.g. click-through rate)



[1] Picture: adapted from https://commons.wikimedia.org/wiki/File:A-B_testing_simple_example.png

A/B testing: frequentist way

- Based on hypothesis testing
- Check whether A and B perform the same or not
- Does not say how much better is A than B

A/B testing: Bayesian approach

- Calculate posterior click-through rates for website layouts A and B and compare them
- Directly calculate the probability that A is better than B
- Quantify how much better it is
- Estimate expected loss in case we make a wrong decision

A/B testing: Bayesian approach

- When a user lands on the website, there are two scenarios:
  - Click (success)
  - No click (failure)
- Use binomial distribution! (probability of success = click rate)

Simulate beta posterior

We know that if the prior is $Beta(a, b)$, then the posterior is $Beta(x, y)$, with:

$$x = \text{NumberOfSuccesses} + a$$

$$y = \text{NumberOfObservations} - \text{NumberOfSuccesses} + b$$

```
def simulate_beta_posterior(trials, beta_prior_a, beta_prior_b):
    num_successes = np.sum(trials)
    posterior_draws = np.random.beta(
      num_successes + beta_prior_a,
      len(trials) - num_successes + beta_prior_b,
      10000
    )
    return posterior_draws
```

Comparing posteriors

Lists of 1s (clicks) and 0s (no clicks):

```
print(A_clicks)
print(B_clicks)
```

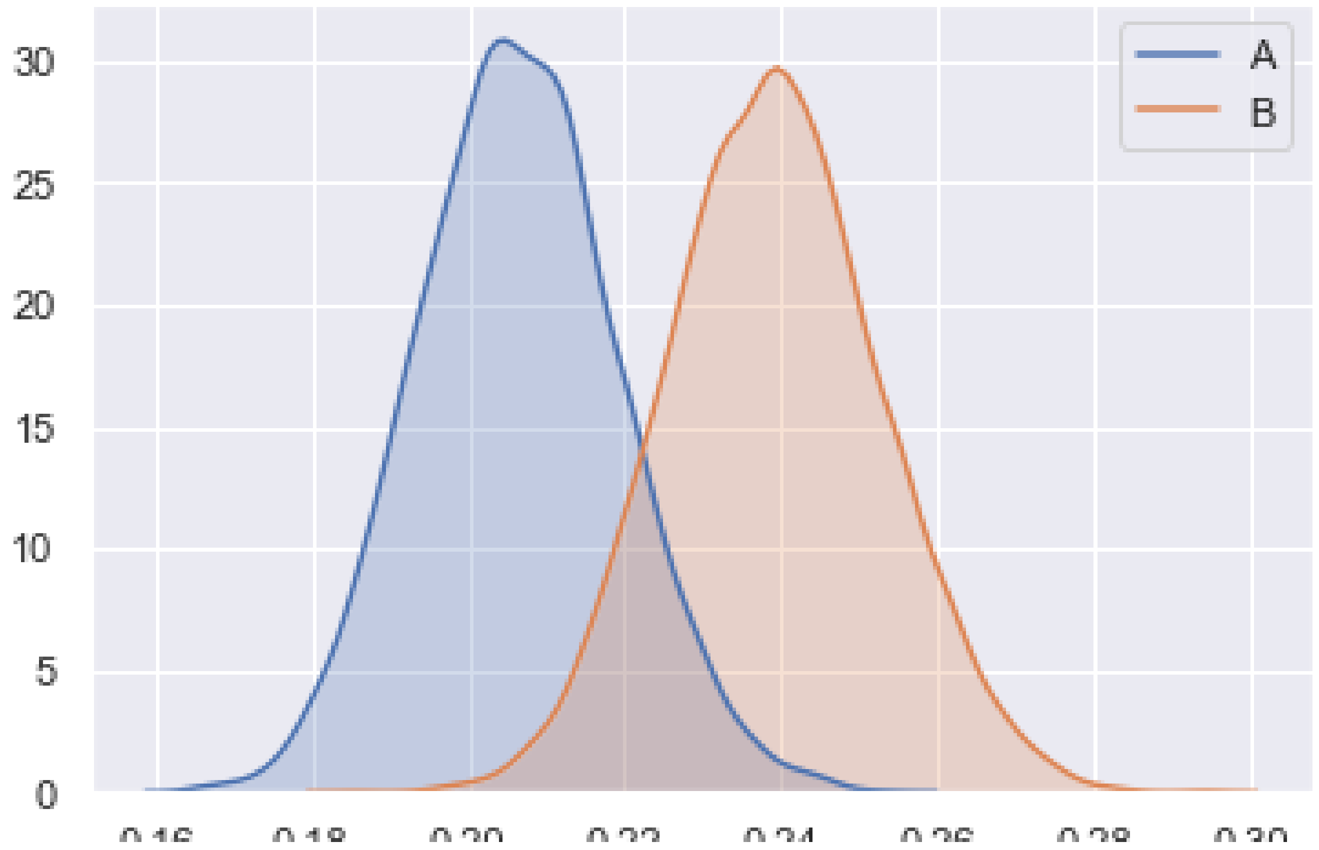
```
[0 1 1 0 0 0 0 0 0 0 1 ... ]
[0 0 0 1 0 0 0 1 1 0 1 ... ]
```

Simulate posterior draws for each layout:

```
A_posterior = simulate_beta_posterior(A_clicks, 1, 1)
B_posterior = simulate_beta_posterior(B_clicks, 1, 1)
```

Plot posteriors:

```
sns.kdeplot(A_posterior, shade=True, label="A")
sns.kdeplot(B_posterior, shade=True, label="B")
plt.show()
```

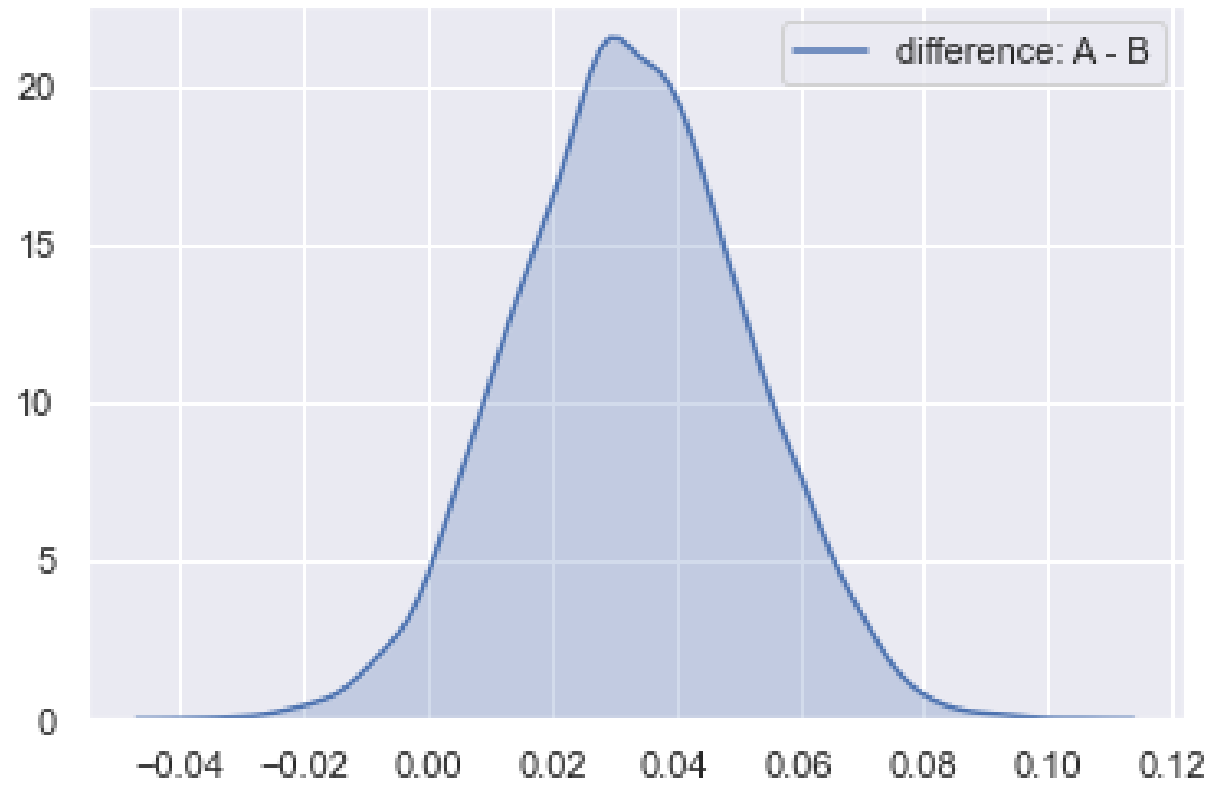


Comparing posteriors

Posterior difference between B and A:

```
diff = B_posterior - A_posterior

sns.kdeplot(diff, shade=True, label="difference: A-B")
plt.show()
```



Probability of B being better:

```
(diff > 0).mean()
```

```
0.9639
```

Expected loss

If we deploy the worse website version, how many clicks do we lose?

```
# Difference (B-A) when A is better
loss = diff[diff < 0]

# Expected (average) loss
expected_loss = loss.mean()
print(expected_loss)
```

```
-0.0077850237030215215
```

Ads data

```
print(ads)
```

```
                              user_id   product site_version                time  banner_clicked
0     f500b9f27ac611426935de6f7a52b71f   clothes      desktop 2019-01-28 16:47:08               0
1     cb4347c030a063c63a555a354984562f  sneakers       mobile 2019-03-31 17:34:59               0
2     89cec38a654319548af585f4c1c76b51   clothes       mobile 2019-02-06 09:22:50               0
3     1d4ea406d45686bdbb49476576a1a985  sneakers       mobile 2019-05-23 08:07:07               0
4     d14b9468a1f9a405fa801a64920367fe   clothes       mobile 2019-01-28 08:16:37               0
...                               ...       ...          ...                 ...             ...
9995  7ca28ccde263a675d7ab7060e9ed0eca   clothes       mobile 2019-02-02 08:19:39               0
9996  7e2ec2631332c6c4527a1b78c7ede789   clothes       mobile 2019-04-04 03:27:05               0
9997  3b828da744e5785f1e67b5df3fda5571   clothes       mobile 2019-04-15 15:59:06               0
9998  6cce0527245bcc8519d698af2224c04a   clothes       mobile 2019-05-21 20:43:21               0
9999  8cf87a02f96327a1a8a93814f34d0d0c  sneakers       mobile 2019-03-02 21:27:57               0
```

Let's A/B test!

BAYESIAN DATA ANALYSIS IN PYTHON

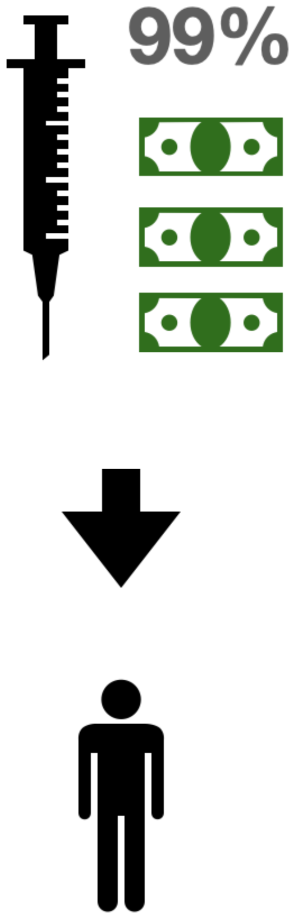
Decision analysis

BAYESIAN DATA ANALYSIS IN PYTHON

Michal Oleszak
Machine Learning Engineer

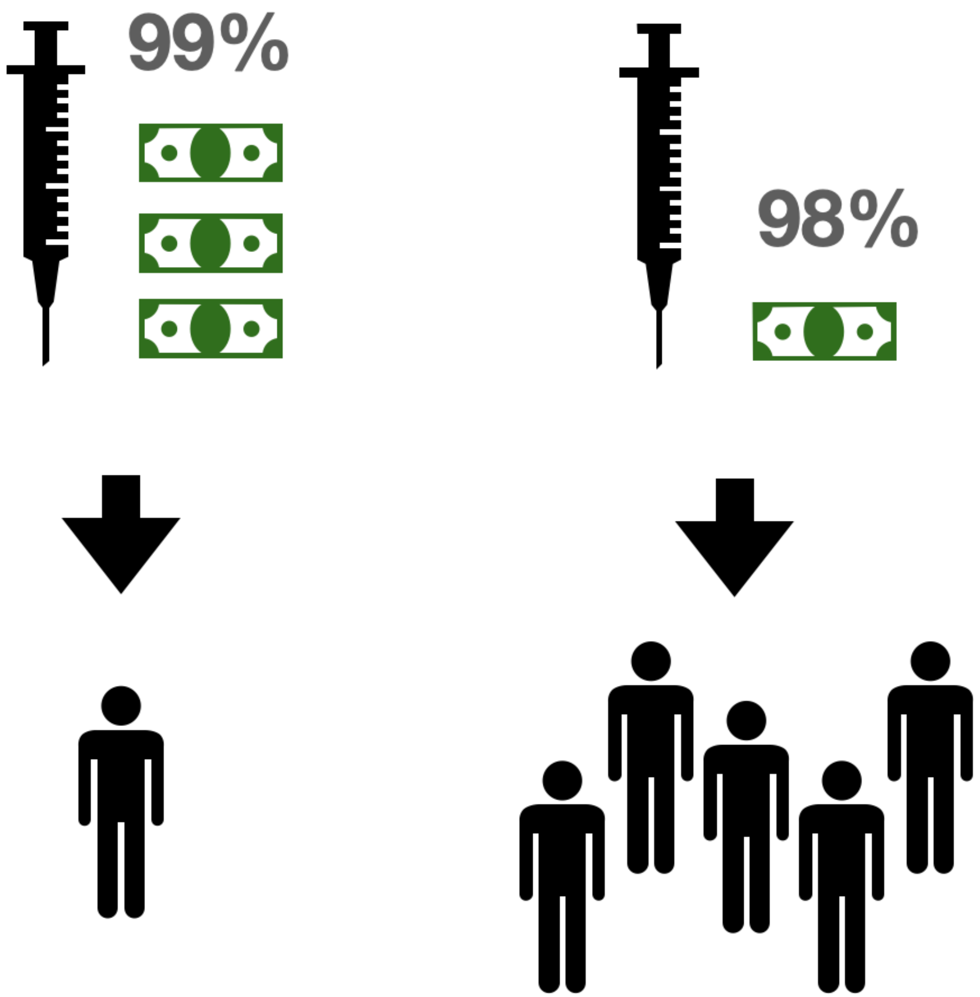
Decision analysis

- Decision-makers care about maximizing profit, reducing costs, saving lives, etc.



Decision analysis

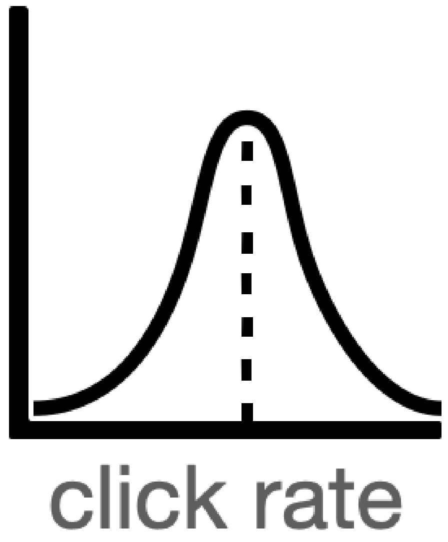
- Decision-makers care about maximizing profit, reducing costs, saving lives, etc.



- Decision analysis → translating parameters to relevant metrics to inform decision-making

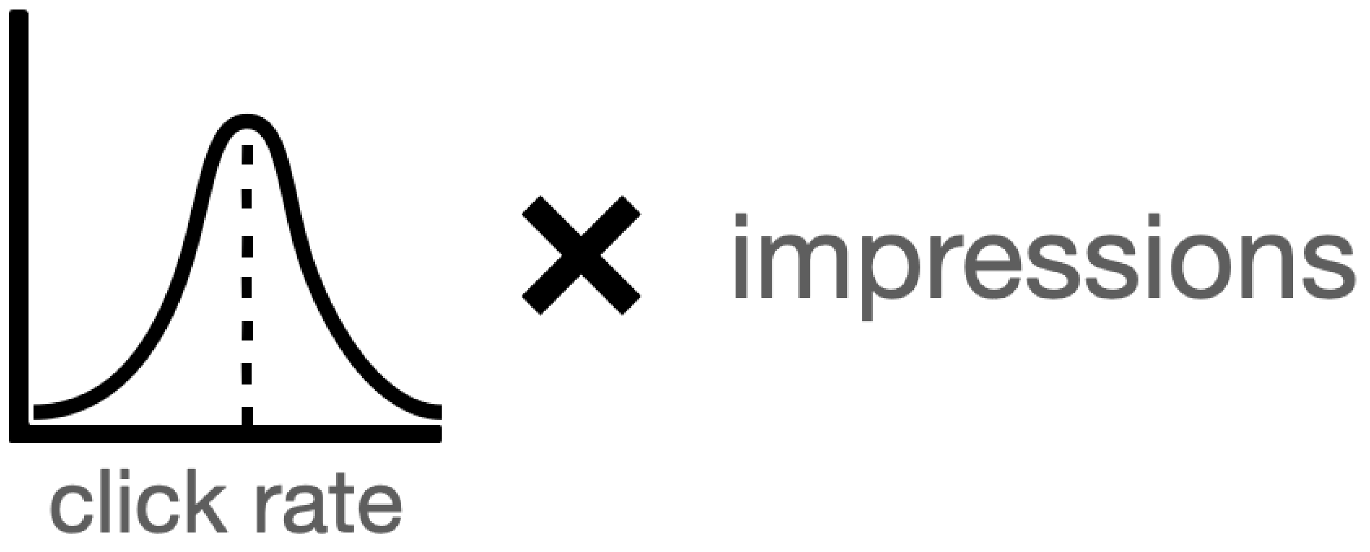
From posteriors to decisions

- To make strategic decisions, one should know the probabilities of different scenarios.
- Bayesian methods allow us to translate parameters into relevant metrics easily.



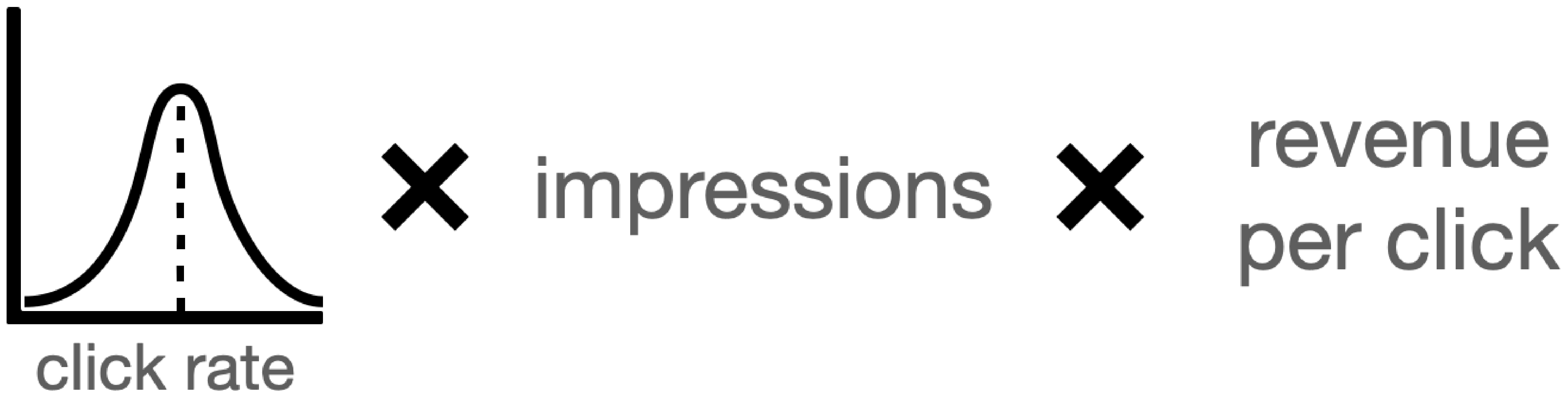
From posteriors to decisions

- To make strategic decisions, one should know the probabilities of different scenarios.
- Bayesian methods allow us to translate parameters into relevant metrics easily.



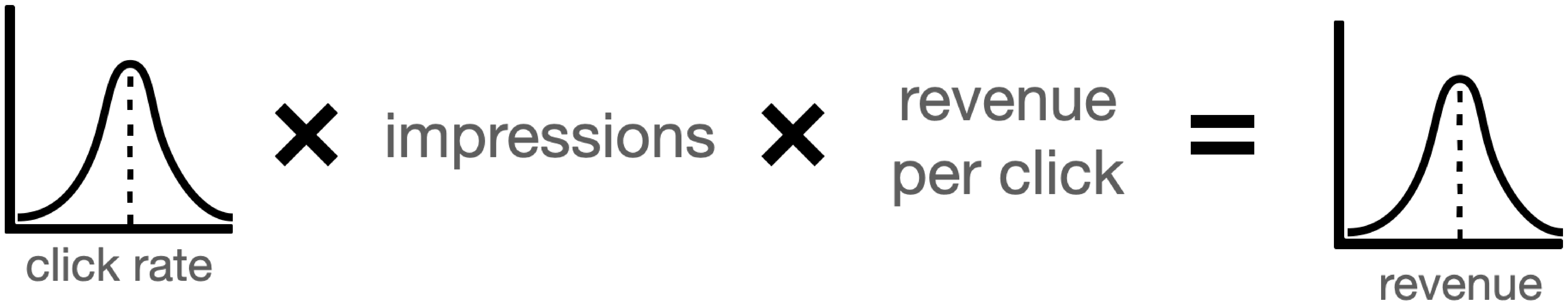
From posteriors to decisions

- To make strategic decisions, one should know the probabilities of different scenarios.
- Bayesian methods allow us to translate parameters into relevant metrics easily.



From posteriors to decisions

- To make strategic decisions, one should know the probabilities of different scenarios.
- Bayesian methods allow us to translate parameters into relevant metrics easily.

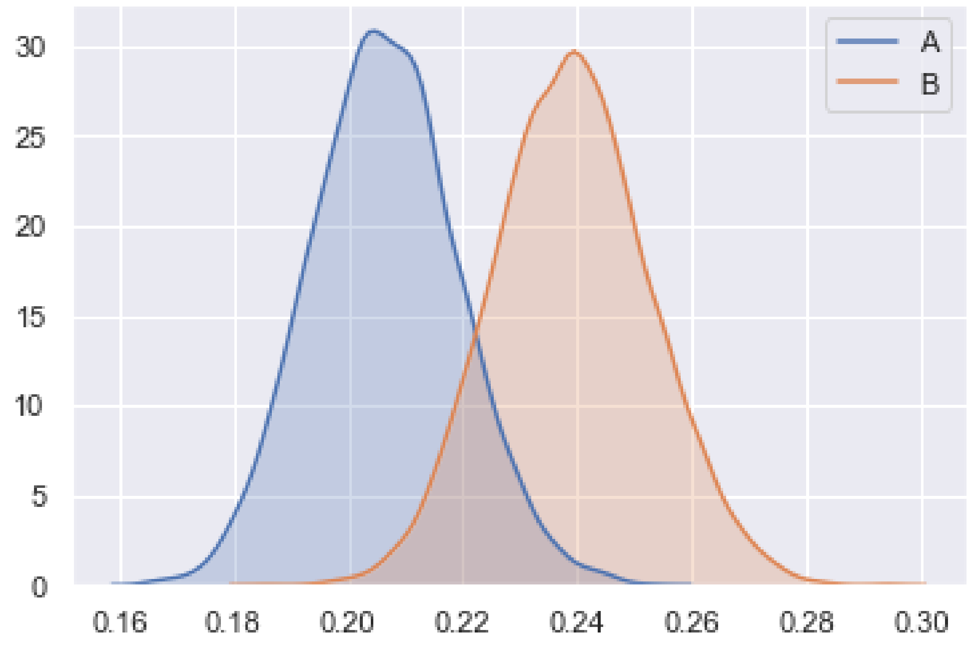


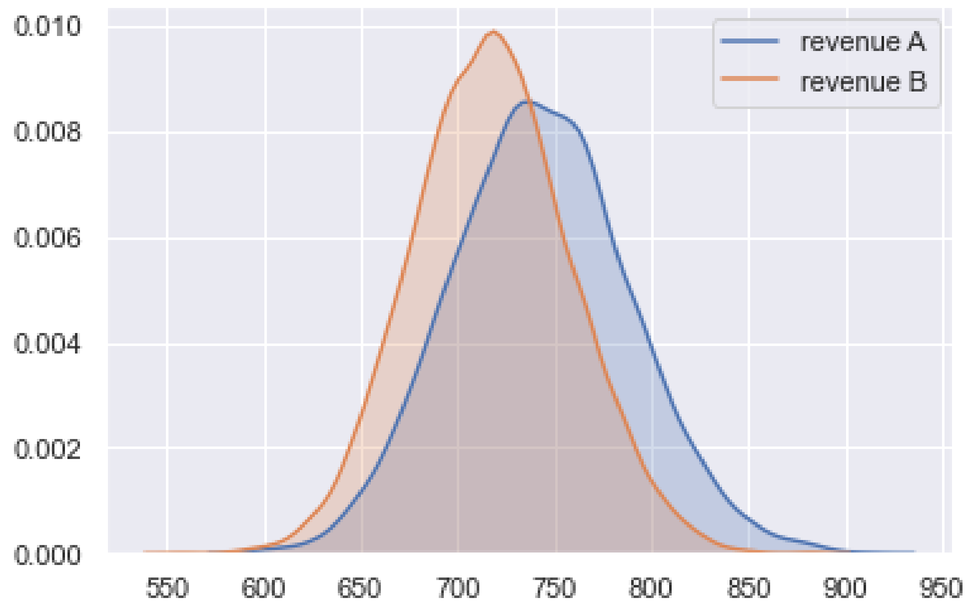
Posterior revenue

```
# Different revenue per click
num_impressions = 1000
rev_per_click_A = 3.6
rev_per_click_B = 3

# Compute number of clicks
num_clicks_A = A_posterior * num_impressions
num_clicks_B = B_posterior * num_impressions

# Compute posterior revenue
rev_A = num_clicks_A * rev_per_click_A
rev_B = num_clicks_B * rev_per_click_B
```



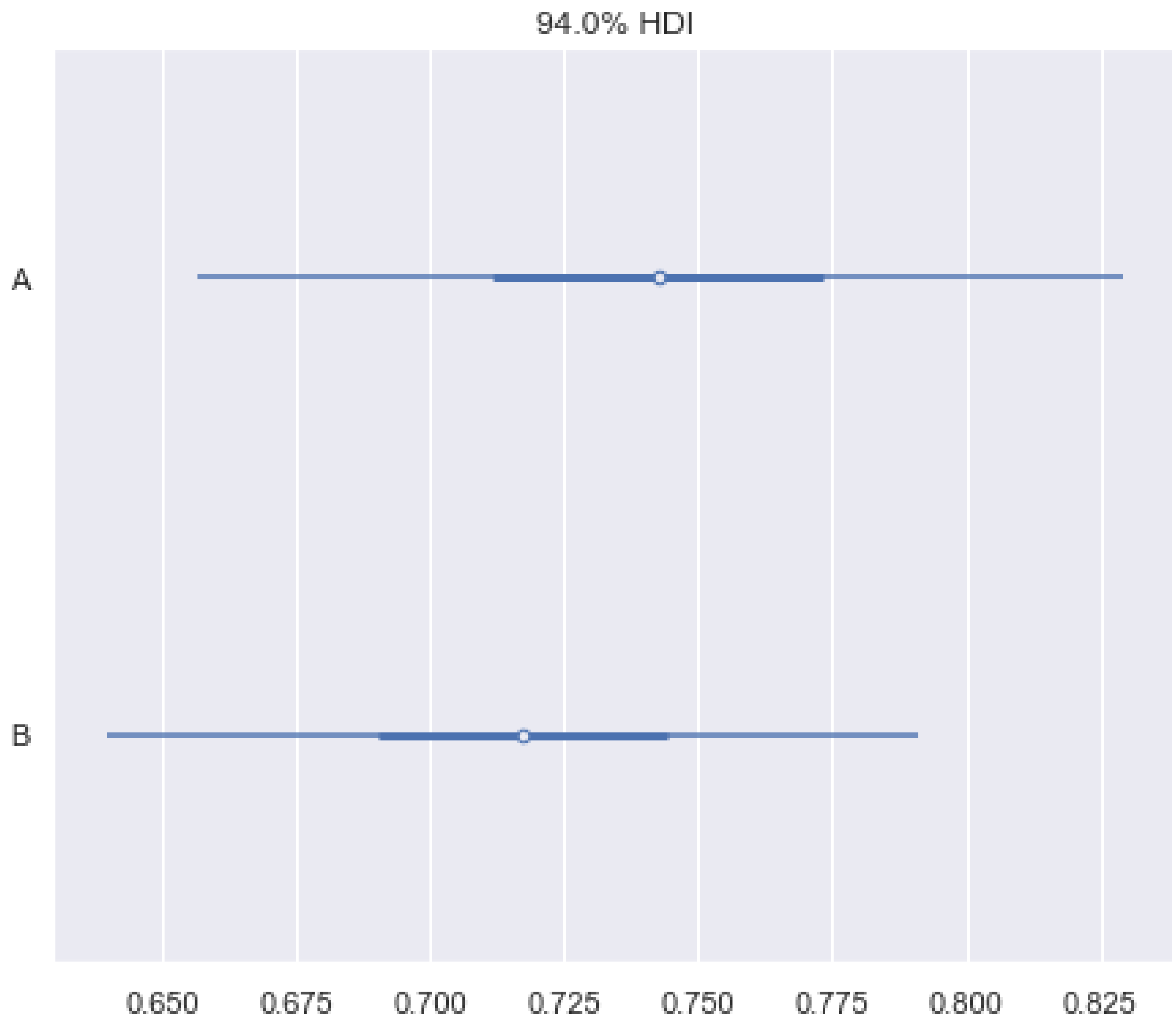


Forest plot

```
import pymc3 as pm

# Collect posterior draws in a dictionary
revenue = {"A": rev_A, "B": rev_B}

# Draw the forest plot
pm.forestplot(revenue)
```

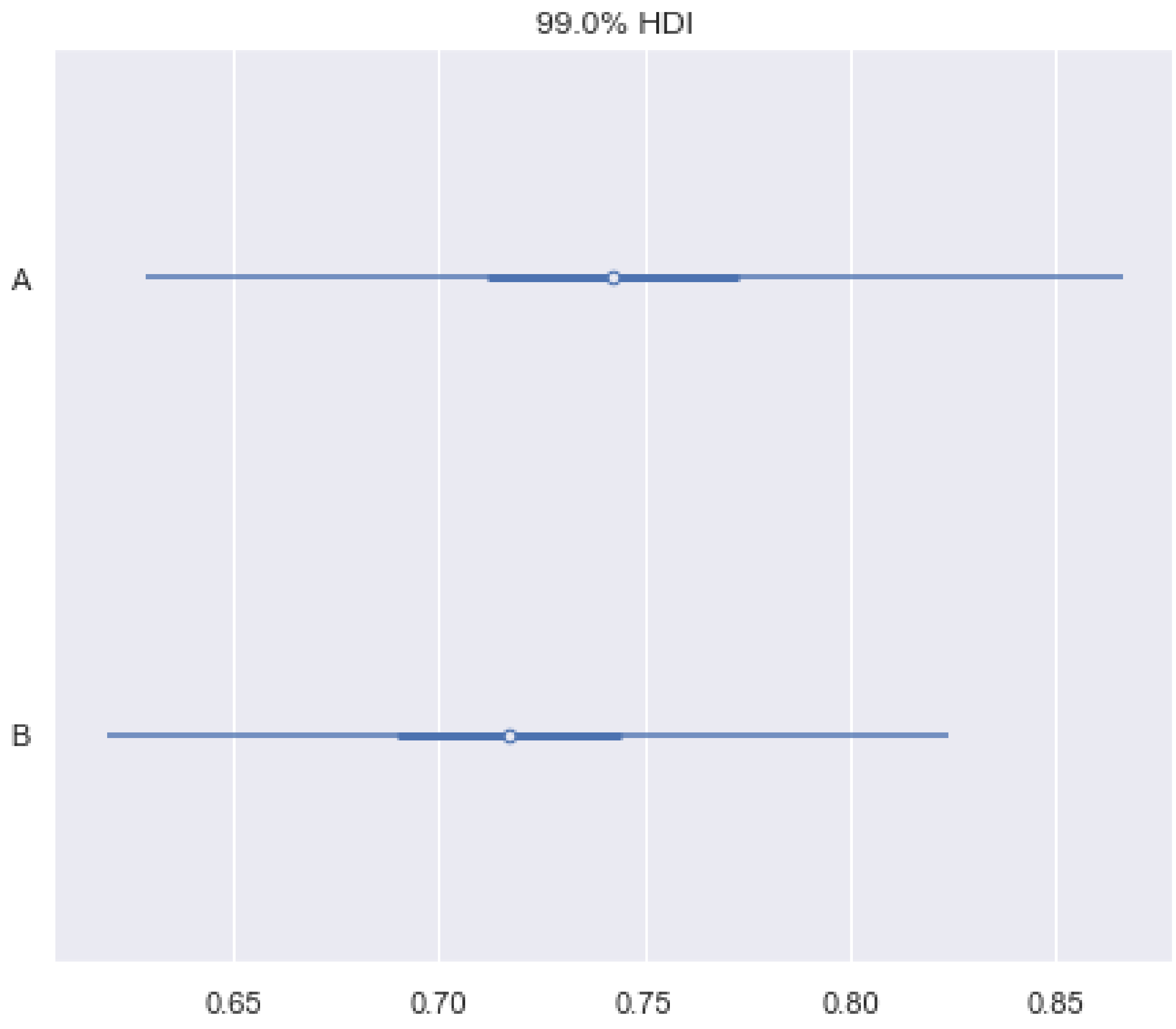


Forest plot

```
import pymc3 as pm

# Collect posterior draws in a dictionary
revenue = {"A": rev_A, "B": rev_B}

# Draw the forest plot
pm.forestplot(revenue, hdi_prob=0.99)
```



Let's analyze decisions!

BAYESIAN DATA ANALYSIS IN PYTHON

Regression and forecasting

BAYESIAN DATA ANALYSIS IN PYTHON

Michal Oleszak
Machine Learning Engineer

Linear regression

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots$$

$$\text{sales} = \beta_0 + \beta_1 \text{marketingSpending}$$

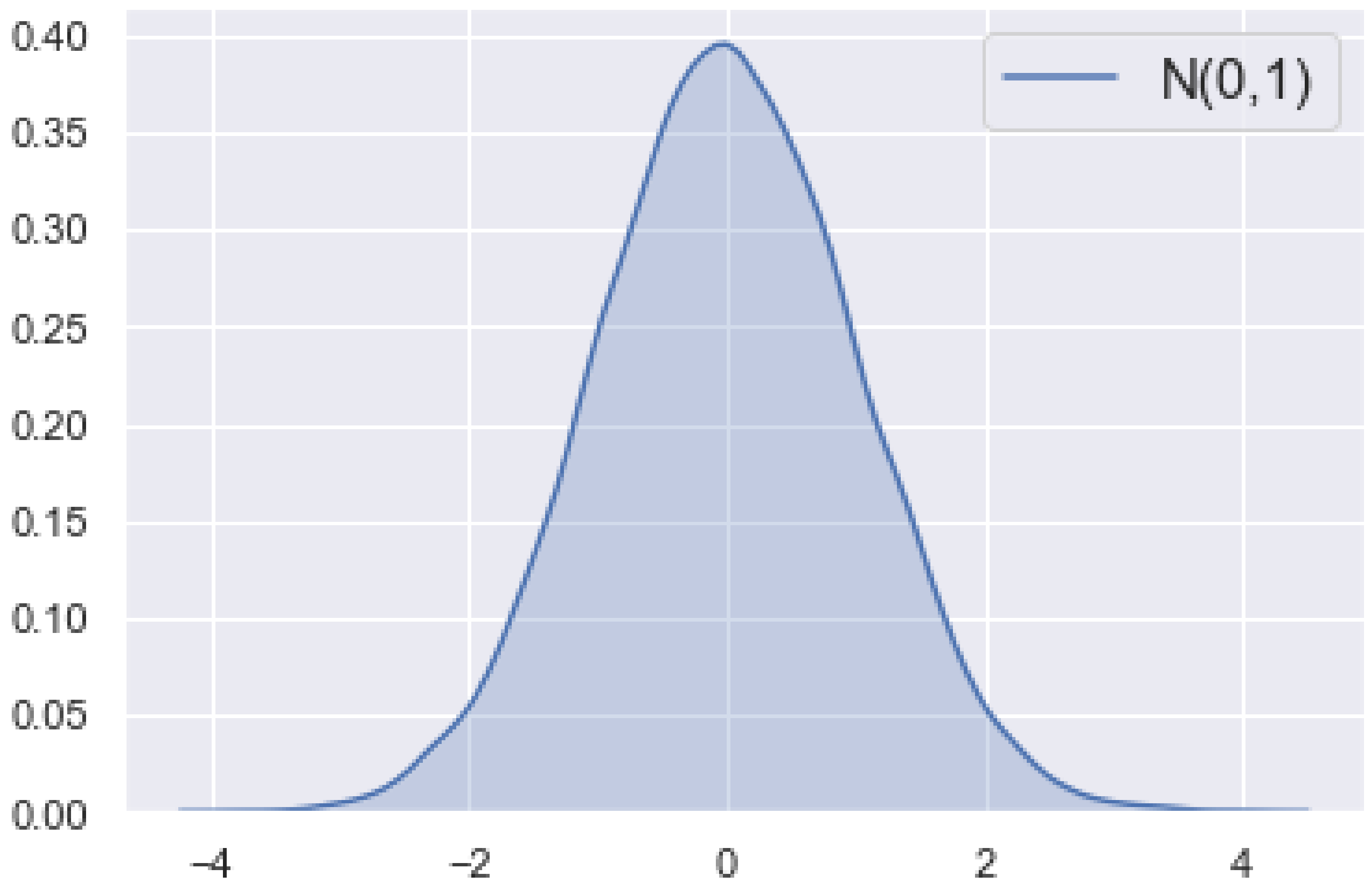
- Frequentist inference:
  - $\text{sales} = \beta_0 + \beta_1 \text{marketingSpending} + \varepsilon$
  - $\varepsilon \sim \mathcal{N}(0, \sigma)$
- Bayesian inference:
  - $\text{sales} \sim \mathcal{N}(\beta_0 + \beta_1 \text{marketingSpending}, \sigma)$

Normal distribution

```
normal_0_1 = np.random.normal(0, 1, size=10000)

sns.kdeplot(normal_0_1, shade=True, label="N(0,1)")

plt.show()
```

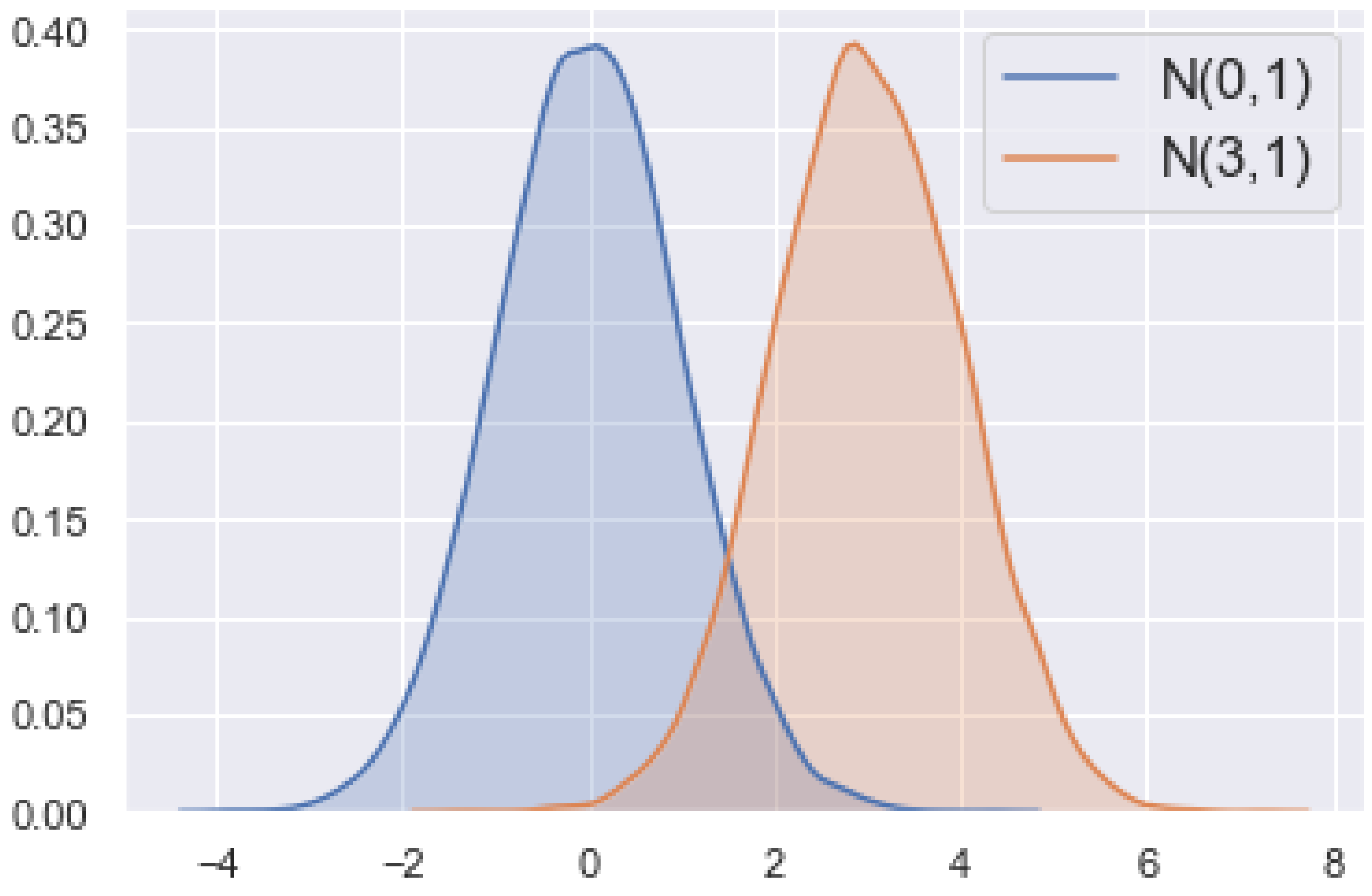


Normal distribution

```
normal_0_1 = np.random.normal(0, 1, size=10000)
normal_3_1 = np.random.normal(3, 1, size=10000)


sns.kdeplot(normal_0_1, shade=True, label="N(0,1)")
sns.kdeplot(normal_3_1, shade=True, label="N(3,1)")


plt.show()
```

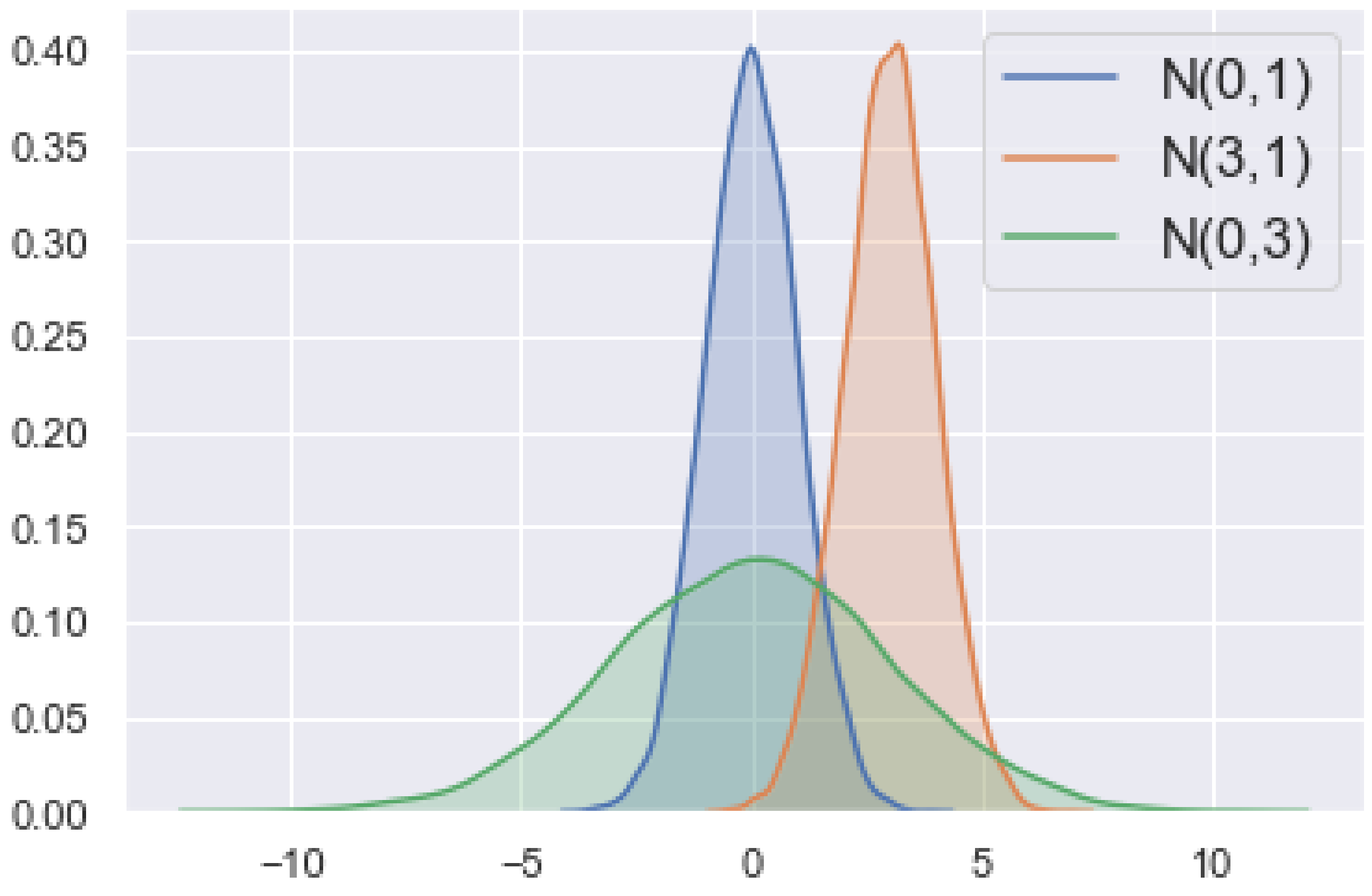


Normal distribution

```
normal_0_1 = np.random.normal(0, 1, size=10000)
normal_3_1 = np.random.normal(3, 1, size=10000)
normal_0_3 = np.random.normal(0, 3, size=10000)

sns.kdeplot(normal_0_1, shade=True, label="N(0,1)")
sns.kdeplot(normal_3_1, shade=True, label="N(3,1)")
sns.kdeplot(normal_0_3, shade=True, label="N(0,3)")

plt.show()
```



Bayesian regression model definition

$$\text{sales} \sim \mathcal{N}(\beta_0 + \beta_1 \text{marketingSpending}, \sigma)$$

$$\beta_0 \sim \mathcal{N}(5, 2)$$

$$\beta_1 \sim \mathcal{N}(2, 10)$$

$$\sigma \sim \mathcal{U}nif(0, 3)$$

- We expect $5000 sales without any marketing.
- We expect $2000 increase in sales from each 1000 increase in spending.
- Uniform prior for standard deviation, as we don't know what it could be.

Estimating regression parameters

- Grid approximation → impractical for many parameters
- Choose conjugate priors and simulate from a known posterior → unintuitive priors
- Third way: simulate from the posterior even with non-conjugate priors!
- For now, assume the parameter draws are given

Plot posterior

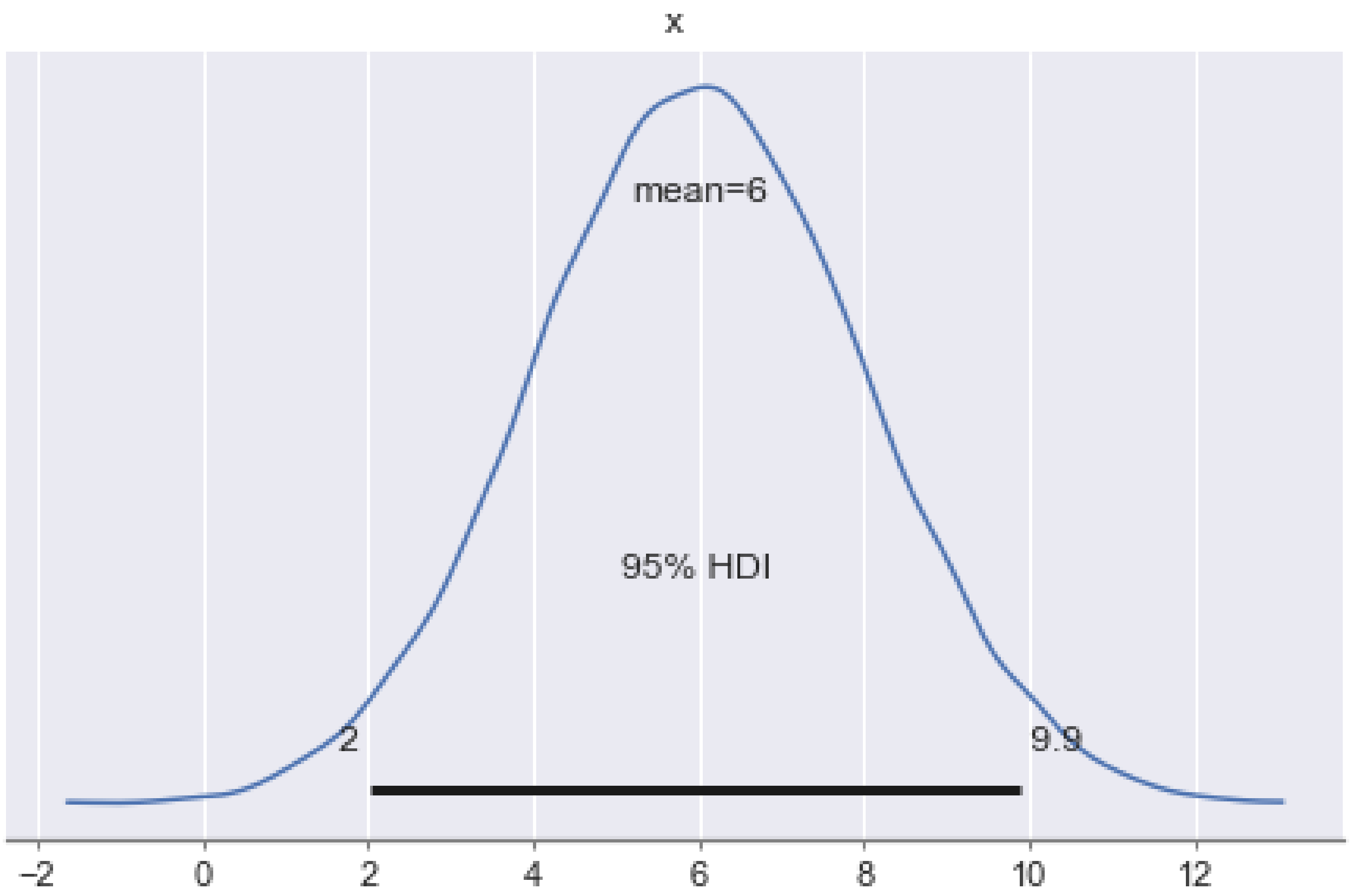
$$\text{sales} = \beta_0 + \beta_1 \text{marketingSpending}$$

```
print(marketing_spending_draws)
```

```
array([9.6153, 8.9922, ..., 4.59565])
```

```
import pymc3 as pm

pm.plot_posterior(
  marketing_spending_draws,
  hdi_prob=0.95
)
```



Posterior draws analysis

```
posterior_draws_df = pd.DataFrame({
    "intercept_draws": intercept_draws,
    "marketing_spending_draws": marketing_spending_draws,
    "sd_draws": sd_draws
})
print(posterior_draws_df)
```

```
       intercept_draws  marketing_spending_draws       sd_draws
count      10000.000000              10000.000000  10000.000000
mean           2.972130                  5.999146      1.337621
std            3.008565                  2.020708      0.471723
min           -8.562093                 -2.842438      0.029643
25%            0.972832                  4.621807      1.003229
50%            3.002940                  5.975067      1.427617
75%            5.020615                  7.362572      1.736310
max           15.228549                 13.258955      1.999834
```

Predictive distribution

How much sales can we expect if we spend $1000 on marketing?

$$\text{sales} \sim \mathcal{N}(\beta_0 + \beta_1 \text{marketingSpending}, \sigma)$$

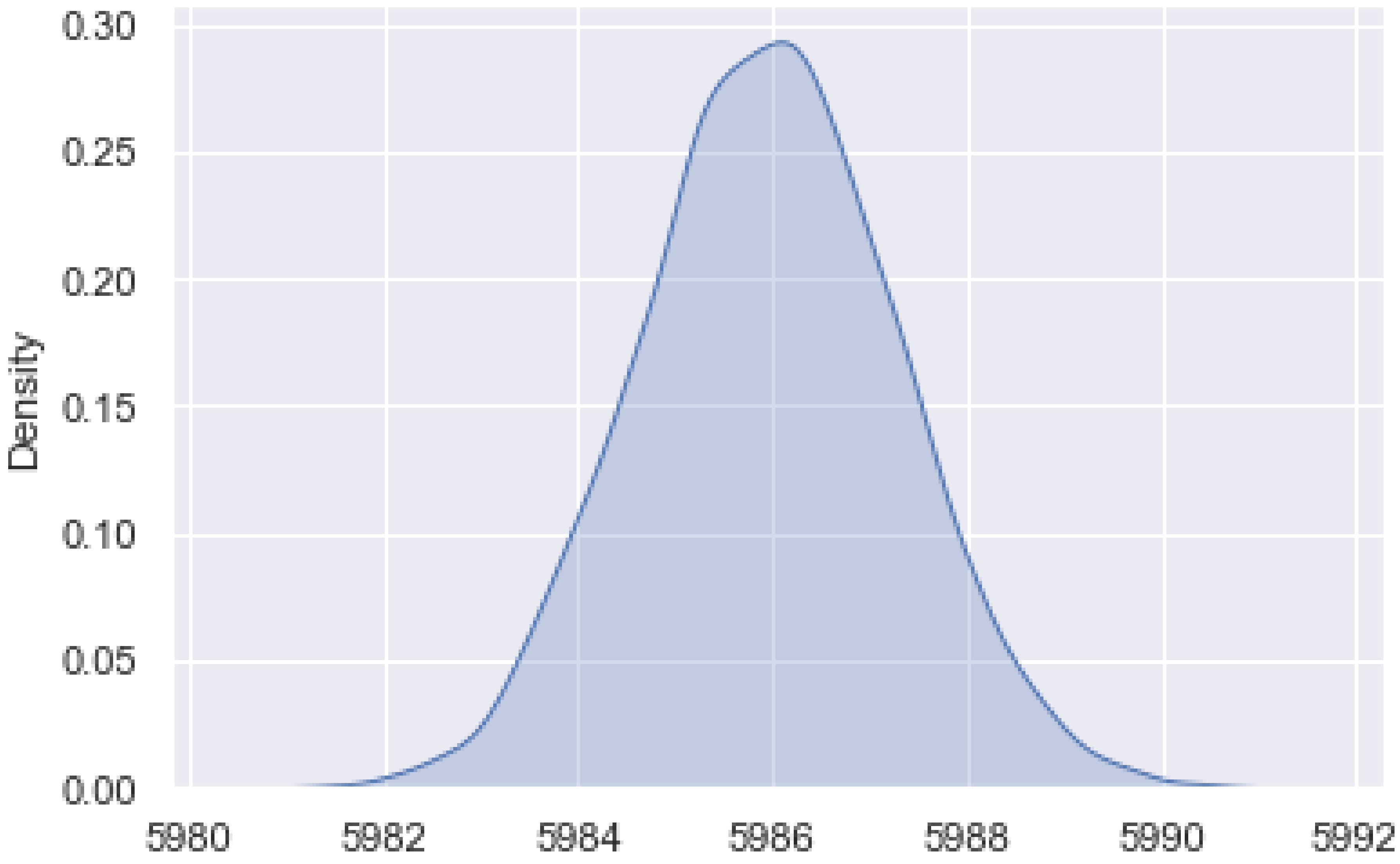
```
# Get point estimates of parameters
intercept_mean = intercept_draws.mean()
marketing_spending_mean = marketing_spending_draws.mean()
sd_mean = sd_draws.mean()

# Calculate mean of predictive distribution
predictive_mean = intercept_mean + marketing_spending_mean * 1000

# Simulate from predictive distribution
prediction_draws = np.random.normal(predictive_mean, sd_mean, size=10000)
```

Predictive distribution

How much sales can we expect if we spend $1000 on marketing?



Let's regress and forecast!

BAYESIAN DATA ANALYSIS IN PYTHON