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

```
import numpy as np
import pandas as pd
import scipy.stats as stats
import matplotlib.pyplot as plt
```

A testing agency wants to analyze the complexity of SAT Exam 2020. They have collected the SAT scores of 1000 students in "sat_score.csv". Let's answer some of the questions that will help to decide the complexity of SAT exam 2020.

1. Calculate the probability that a student will score less than 800 in SAT exam
2. Calculate the probability that a student will score more than 1300 in SAT exam
3. Calculate the minimum marks a student must score in order to secure 90th percentile
4. Calculate the minimum marks a student must score in order to be in the top 5%

```
# Given data
sat_mean = 1000
sat_std_dev = 200
sat_highest_score = 1350

act_mean = 20
act_std_dev = 5
act_highest_score = 30
```

```
# Load the dataset
sat_scores_data = pd.read_csv('/content/sat score.csv')
```

```
# Extract SAT scores
sat_scores = sat_scores_data['score']
```

```
# 1. Analyze SAT Exam complexity
# Q1. Probability that a student scores less than 800
prob_less_800 = np.mean(sat_scores < 800)
```

```
# Q2. Probability that a student scores more than 1300
prob_more_1300 = np.mean(sat_scores > 1300)
```

```
# Q3. Minimum marks to secure 90th percentile
percentile_90 = np.percentile(sat_scores, 90)
```

```
# Q4. Minimum marks to be in top 5%
percentile_95 = np.percentile(sat_scores, 95)
```

```
# Display results
print(f"Probability that a student scores less than 800: {prob_less_800}")
print(f"Probability that a student scores more than 1300: {prob_more_1300}")
print(f"Minimum marks to secure 90th percentile: {percentile_90}")
print(f"Minimum marks to secure top 5%: {percentile_95}")
```

```
Probability that a student scores less than 800: 0.157
Probability that a student scores more than 1300: 0.068
Minimum marks to secure 90th percentile: 1269.0
Minimum marks to secure top 5%: 1338.1
```

Suppose we know that the SAT scores are normally distributed with mean 1000 and standard deviation 200 and ACT scores are normally distributed with mean 20 and standard deviation 5.

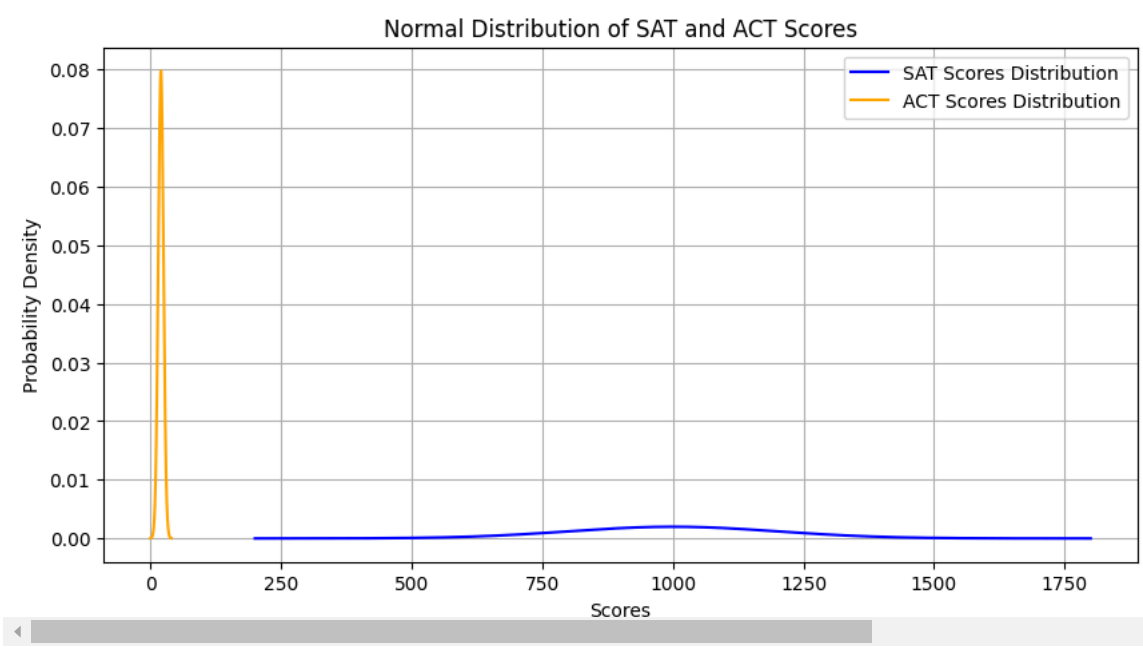
A college provides admission only on the basis of SAT and ACT scores. The college admin decides to give the top performer fellowship to the student who has performed the best among all applicants. The highest score received from applicants who appeared for SAT is 1350 and the highest score received from applicants who appeared for ACT is 30.

Help the college to choose the best candidate for the fellowship!

```
# 2. Compare SAT and ACT top scores
# SAT details
sat_mean = 1000
sat_std = 200
sat_top_score = 1350


# ACT details
act_mean = 20
act_std = 5
act_top_score = 30


# Z-scores for SAT and ACT top performers
z_sat = (sat_top_score - sat_mean) / sat_std
z_act = (act_top_score - act_mean) / act_std


# Display Z-scores
print(f"Z-score for SAT top performer: {z_sat}")
print(f"Z-score for ACT top performer: {z_act}")
```

```
Z-score for SAT top performer: 1.75
Z-score for ACT top performer: 2.0
```

```
# Convert z-scores to probabilities
prob_sat = norm.cdf(z_sat)
prob_act = norm.cdf(z_act)


# Determine the top performer
if prob_sat > prob_act:
    best_candidate = "SAT candidate"
else:
    best_candidate = "ACT candidate"


# 3. Visualize normal distribution for SAT and ACT
x_sat = np.linspace(sat_mean - 4*sat_std, sat_mean + 4*sat_std, 1000)
y_sat = stats.norm.pdf(x_sat, sat_mean, sat_std)


x_act = np.linspace(act_mean - 4*act_std, act_mean + 4*act_std, 1000)
y_act = stats.norm.pdf(x_act, act_mean, act_std)


plt.figure(figsize=(10, 5))
plt.plot(x_sat, y_sat, label="SAT Scores Distribution", color='blue')
plt.plot(x_act, y_act, label="ACT Scores Distribution", color='orange')
plt.title("Normal Distribution of SAT and ACT Scores")
plt.xlabel("Scores")
plt.ylabel("Probability Density")
plt.legend()
plt.grid(True)
plt.show()
```




```
# Conclusion: Based on Z-scores, determine the top performer
if z_sat > z_act:
    print("The SAT top performer should receive the fellowship.")
else:
    print("The ACT top performer should receive the fellowship.")
```

```
The ACT top performer should receive the fellowship.
```

```
# Print results
print(f"Z-score for SAT highest score: {z_sat:.3f}")
print(f"Percentile for SAT highest score: {prob_sat * 100:.2f}%")
print(f"Z-score for ACT highest score: {z_act:.3f}")
print(f"Percentile for ACT highest score: {prob_act * 100:.2f}%")
print(f"The best candidate for the fellowship is the {best_candidate}.")
```

```
Z-score for SAT highest score: 1.750
Percentile for SAT highest score: 95.99%
Z-score for ACT highest score: 2.000
Percentile for ACT highest score: 97.72%
The best candidate for the fellowship is the ACT candidate.
```

80% of all the visitors to Lavista Museum end up buying souvenirs from the souvenir shop at the Museum. On the coming Sunday, if a random sample of 10 visitors is picked:

- 1. Find the probability that every visitor will end up buying from the souvenir shop
- 2. Find the probability that a maximum of 7 visitors will buy souvenirs from the souvenir shop

Let's check first whether we satisfy the assumptions of the binomial distribution.

- There are only two possible outcomes (success or failure) for each trial. A visitor will buy souvenirs from the souvenir shop or not (yes or no).
- Number of trials (n) is fixed - There are 10 visitors in the sample.
- Each trial is independent of the other trials - It is reasonable to assume that the buying activity of visitors is independent.
- The probability of success (p) is the same for each trial - The probability of success for each visitor is 0.8.

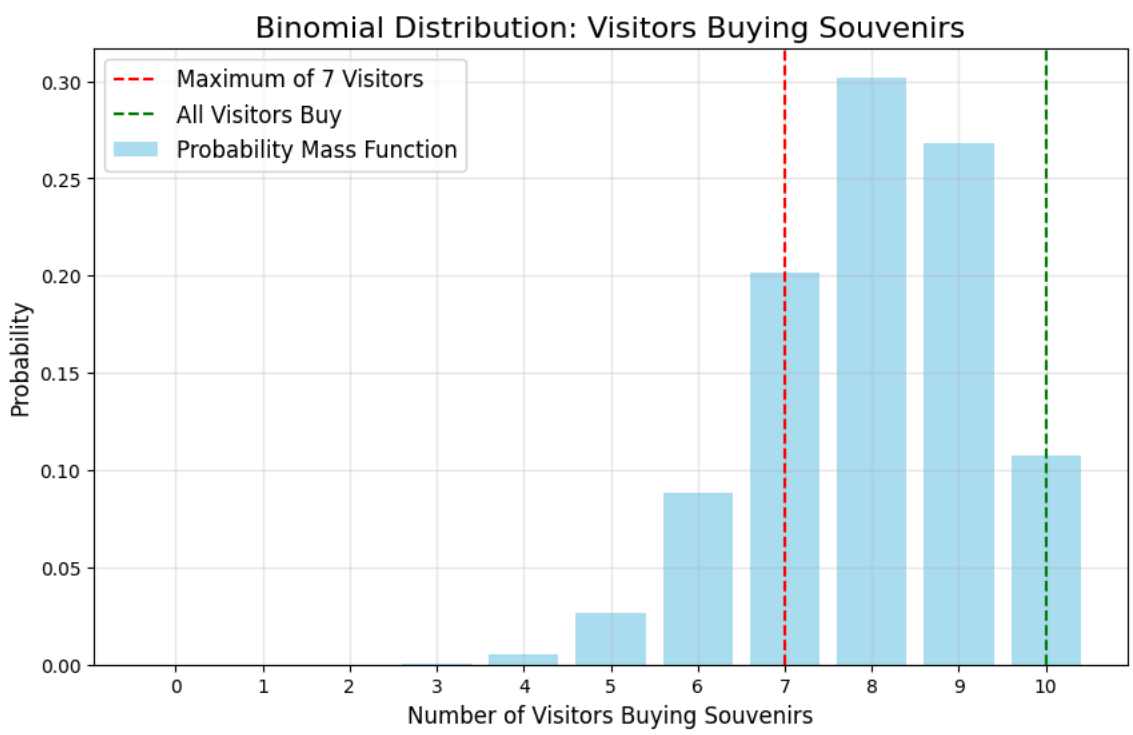
```
from scipy.stats import binom
```

```
# Given data
n = 10  # Number of trials (visitors)
p = 0.8  # Probability of success (buying souvenirs)
```

```
# 1. Probability that every visitor will end up buying from the souvenir shop
prob_all_buy = binom.pmf(n, n, p)



# 2. Probability that a maximum of 7 visitors will buy souvenirs
prob_max_7 = binom.cdf(7, n, p)


print(f"1. Probability that every visitor will buy souvenirs: {prob_all_buy:.4f}")
print(f"2. Probability that a maximum of 7 visitors will buy souvenirs: {prob_max_7:.4f}")
```

```
1. Probability that every visitor will buy souvenirs: 0.1074
2. Probability that a maximum of 7 visitors will buy souvenirs: 0.3222
```

```
# Generate data for visualization
x = np.arange(0, n + 1)
y = binom.pmf(x, n, p)


# Plot the binomial distribution
plt.figure(figsize=(10, 6))
plt.bar(x, y, color='skyblue', alpha=0.7, label='Probability Mass Function')
plt.axvline(x=7, color='red', linestyle='--', label='Maximum of 7 Visitors')
plt.axvline(x=10, color='green', linestyle='--', label='All Visitors Buy')
plt.title('Binomial Distribution: Visitors Buying Souvenirs', fontsize=16)
plt.xlabel('Number of Visitors Buying Souvenirs', fontsize=12)
plt.ylabel('Probability', fontsize=12)
plt.xticks(range(0, n + 1))
plt.legend(fontsize=12)
plt.grid(alpha=0.3)
plt.show()
```



Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.