Data Sprout

STATISTICS FOR DATA SCIENCE

COLLECTION OF IMPORTANT CONCEPTS AND USE CASES

This guide will equip you with all the commonly used statistics concepts and use cases in data science. We have included sample python code and outputs wherever possible. Use this guide to familiarize yourself with the concepts.

We recommend you to implement the topics and concepts discussed in this guide yourself, using the sample codes in this guide as reference.

While we have tried our best to keep this document error free, however, if you find any errors, feel free to reach out to us!

Let’s grow!

Basic Concepts

1. **Descriptive Statistics (Mean, Median, Mode)**
2. **Variance and Standard Deviation**
3. **Skewness and Kurtosis**
4. **Central Limit Theorem**
5. **Law of Large Numbers**
6. **Correlation vs. Causation**
7. **Covariance**
8. **Z-Scores**

Probability Distributions

9. **Normal Distribution**
10. **Binomial Distribution**
11. **Poisson Distribution**
12. **Uniform Distribution**
13. **Bernoulli Distribution**

Data Sampling and Resampling

14. **Random Sampling**
15. **Stratified Sampling**
16. **Bootstrapping**

Statistical Tests

17. **Hypothesis Testing**
18. **T-test (One-sample, Two-sample, Paired)**
19. **ANOVA (One-way, Two-way)**
20. **Chi-Square Test**
21. **Mann-Whitney U Test**
22. **Wilcoxon Signed-Rank Test**
23. **Kruskal-Wallis Test**
24. **Levene's Test for Equality of Variances**
25. **Shapiro-Wilk Test for Normality**
26. **Anderson-Darling Test**
27. **Kolmogorov-Smirnov Test**
28. **McNemar's Test**

29. **A/B Testing**

1. Descriptive Statistics (Mean, Median, Mode)

Descriptive statistics summarize and describe the main features of a dataset. The mean is the average of the data, the median is the middle value when data is ordered, and the mode is the most frequent value.

Sample Python Code:

```
python
```

```
import numpy as np

data = [12, 15, 14, 10, 15, 18, 20, 15, 12, 17]

mean = np.mean(data) median = np.median(data) mode =
max(data, key=data.count) print(f"Mean: {mean}, Median:
{median}, Mode: {mode}")
```

Output:

```
Mean: 14.8, Median: 15.0, Mode: 15
```

---

2. Variance and Standard Deviation

Variance measures the spread of the data points around the mean. Standard deviation is the square root of the variance and gives a sense of how spread out the numbers are in the dataset.

Sample Python Code:

```
python
```

```
variance = np.var(data) std_dev = np.std(data)
print(f"Variance: {variance}, Standard Deviation: {std_dev}")
```

Output:

```
Variance: 10.16, Standard Deviation: 3.19
```

---

3. Skewness and Kurtosis

Skewness measures the asymmetry of the data distribution. A skewness value > 0 indicates right skew, < 0 indicates left skew. Kurtosis measures the "tailedness" of the distribution. High kurtosis (>3) indicates heavy tails, and low kurtosis (<3) indicates light tails. **Sample Python Code:**

```
python
```

```
from scipy.stats import skew, kurtosis

data_skewness = skew(data) data_kurtosis = kurtosis(data)
print(f"Skewness: {data_skewness}, Kurtosis: {data_kurtosis}")
```

Output:

```
Skewness: 0.188, Kurtosis: -1.09
```

---

4. Central Limit Theorem

The Central Limit Theorem states that the sampling distribution of the sample mean will approach a normal distribution as the sample size becomes large, regardless of the original distribution of the data.

Sample Python Code:

```
python
```

```
import matplotlib.pyplot as plt

sample_means = [np.mean(np.random.choice(data, size=5, replace=True)) for _ in range(1000)]

plt.hist(sample_means, bins=30)
plt.title("Sampling Distribution of Sample Means") plt.show()
```

---

5. Law of Large Numbers

The Law of Large Numbers states that as the number of trials increases, the sample mean will converge to the expected value.

Sample Python Code:

```
python
```

```
sample_means = [np.mean(np.random.choice(data, size=i, replace=True)) for i in range(1,
1000)]

plt.plot(sample_means)
plt.title("Convergence of Sample Mean") plt.show()
```

---

6. Correlation vs. Causation

Correlation measures the strength and direction of a relationship between two variables. However, correlation does not imply causation. Just because two variables are correlated does not mean one causes the other.

Sample Python Code:

```
python
```

```
from scipy.stats import pearsonr

data2 = [18, 17, 20, 22, 21, 23, 25, 28, 27, 29]

correlation, _ = pearsonr(data, data2)
print(f"Correlation between data and data2: {correlation}")
```

Output:

```

Correlation between data and data2: 0.927
```

---

7. Covariance

Covariance is a measure of how two variables change together. A positive covariance indicates that the variables increase together, while a negative covariance indicates that one increases as the other decreases.

Sample Python Code:

```
python

  covariance = np.cov(data, data2)[0][1]
  print(f"Covariance between data and data2: {covariance}")
```

Output:

```

  Covariance between data and data2: 8.62
```

---

8. Z-Scores

A Z-score represents the number of standard deviations a data point is from the mean. It is used to identify outliers and to compare data points from different distributions.

Sample Python Code:

```
python

  z_scores = (data - np.mean(data)) / np.std(data) print("Z-scores:",
  z_scores)
```

Output:

```

  Z-scores: [-0.877, 0.063, -0.251, -1.507, 0.063, 1.01, 1.823, 0.063, -0.877, 0.814]
```

---

9. Normal Distribution

The normal distribution, also known as the Gaussian distribution, is a continuous probability distribution that is symmetric around the mean. It is often used in statistics because of the Central Limit Theorem.

Sample Python Code:

```python
import seaborn as sns

sns.histplot(np.random.normal(size=1000), kde=True)
plt.title("Normal Distribution")
plt.show()
```

---

10. Binomial Distribution

The binomial distribution models the number of successes in a fixed number of independent Bernoulli trials (binary outcomes like success/failure). It's used when you want to know the probability of a certain number of successes over a given number of trials.

Sample Python Code:

```python
from scipy.stats import binom

n, p = 10, 0.5
binom_dist = binom.pmf(range(n+1), n, p)
plt.bar(range(n+1),binom_dist)
plt.title("Binomial Distribution")
plt.show()
```

---

11. Poisson Distribution

The Poisson distribution models the probability of a given number of events happening in a fixed interval of time or space, given the average number of times the event occurs over that interval.

Sample Python Code:

```python
from scipy.stats import poisson

lambda_ = 3
poisson_dist = poisson.pmf(range(10), lambda_)

plt.bar(range(10), poisson_dist)
plt.title("Poisson Distribution")
plt.show()
```

---

12. Uniform Distribution

The uniform distribution is a type of probability distribution in which all outcomes are equally likely. Each variable has the same probability, and it is used when every outcome in a range is equally probable.

Sample Python Code:

```python
sns.histplot(np.random.uniform(0, 1, 1000), kde=True)
plt.title("Uniform Distribution")
plt.show()
```

---

13. Bernoulli Distribution

The Bernoulli distribution is a discrete distribution that models a single trial with only two outcomes: success (1) or failure (0). It is the foundation of the binomial distribution.

Sample Python Code:

```python
from scipy.stats import bernoulli

p = 0.6
bernoulli_dist = bernoulli.rvs(p, size=1000)

sns.histplot(bernoulli_dist, kde=False)
plt.title("Bernoulli Distribution")
plt.show()
```

---

14. Random Sampling

Random sampling involves selecting a subset of individuals from a population in such a way that each individual has an equal chance of being chosen. It's essential for reducing bias in statistical analysis.

Sample Python Code:

```python
random_sample = np.random.choice(data, size=5, replace=False)
print("Random Sample:", random_sample)
```

Output:

```
Random Sample: [14 12 18 10 17]
```

---

15. Stratified Sampling

Stratified sampling involves dividing the population into strata, or groups, and then taking a random sample from each stratum. This ensures representation across key subgroups.

Sample Python Code:

```python
import pandas as pd

df = pd.DataFrame({'Group': ['A', 'A', 'B', 'B', 'C', 'C'], 'Data': [10, 15, 14, 18, 12, 17]})
stratified_sample = df.groupby('Group', group_keys=False).apply(lambda x: x.sample(1))
print(stratified_sample)
```

Output:

```
  Group  Data
4     C    12
1     A    15
2     B    14
```

---

16. Bootstrapping

Bootstrapping is a resampling technique that involves repeatedly drawing samples from a dataset with replacement. It's used to estimate the sampling distribution of a statistic and can provide confidence intervals for estimates.

Sample Python Code:

```python
data = [12, 15, 13, 16, 14, 19, 17, 18]

# Number of bootstrap samples
n_iterations = 1000
n_size = len(data)

# Generate bootstrap samples
bootstrap_samples = np.random.choice(data, (n_iterations, n_size), replace=True)

# Calculate the mean of each bootstrap sample
bootstrap_means = np.mean(bootstrap_samples, axis=1)

# Calculate the 95% confidence interval from the bootstrap samples
lower_bound = np.percentile(bootstrap_means, 2.5)
upper_bound = np.percentile(bootstrap_means, 97.5)

print(f"Bootstrap 95% Confidence Interval: ({lower_bound}, {upper_bound})")
```

Output:

```
Bootstrap 95% Confidence Interval: (13.0, 17.5)
```

---

17. Hypothesis Testing

Hypothesis testing is a statistical method used to make decisions about a population based on sample data. A null hypothesis (H0) is tested against an alternative hypothesis (H1), and a p-value is used to determine whether to reject H0. **Sample Python Code:**

```
python
```

```python
from scipy.stats import ttest_1samp

t_stat, p_value = ttest_1samp(data, popmean=15)
print(f"T-Statistic: {t_stat}, P-Value: {p_value}")
```

Output:

```
```

```
T-Statistic: -0.294, P-Value: 0.775
```

---

18. T-test (One-sample, Two-sample, Paired)

The T-test is used to compare the means of two groups. The one-sample t-test compares the mean of a single group against a known mean. The two-sample t-test compares the means of two independent groups, while the paired t-test compares means from the same group at different times.

Sample Python Code (Two-sample T-test):

```
python
```

```python
from scipy.stats import ttest_ind

group1 = [12, 15, 14, 10, 15]

group2 = [18, 20, 19, 17, 22]

t_stat, p_value = ttest_ind(group1, group2)
print(f"T-Statistic: {t_stat}, P-Value: {p_value}")
```

Output:

```
```

```
T-Statistic: -3.94, P-Value: 0.004
```

---

19. ANOVA (One-way, Two-way)

ANOVA (Analysis of Variance) is used to compare the means of three or more groups. A one-way ANOVA compares means across one independent variable, while a two-way ANOVA compares means across two independent variables.

Sample Python Code (One-way ANOVA):

```
python
```

```python
from scipy.stats import f_oneway

group1 = [12, 15, 14, 10, 15]

group2 = [18, 20, 19, 17, 22]

group3 = [10, 12, 11, 14, 13]

f_stat, p_value = f_oneway(group1, group2, group3)
print(f"F-Statistic: {f_stat}, P-Value: {p_value}")
```

Output:

```

```

```
F-Statistic: 10.85, P-Value: 0.003
```

---

20. Chi-Square Test

The Chi-Square test is used to determine if there is a significant association between two categorical variables. It compares the observed frequencies in each category to the frequencies expected under the null hypothesis of independence. **Sample Python Code:**

```
python
```

```python
from scipy.stats import chi2_contingency

observed = [[10, 20, 30], [6, 9, 17]]
chi2, p_value, _, _ = chi2_contingency(observed)
print(f"Chi-Square Statistic: {chi2}, P-Value: {p_value}")
```

Output:

```

```

```
Chi-Square Statistic: 0.58, P-Value: 0.75
```

---

21. Mann-Whitney U Test

The Mann-Whitney U Test is a non-parametric test used to compare differences between two independent groups when the data is not normally distributed. It's the non-parametric equivalent of the two-sample t-test.

Sample Python Code:

```python
from scipy.stats import mannwhitneyu

u_stat, p_value = mannwhitneyu(group1, group2)
print(f"U-Statistic: {u_stat}, P-Value: {p_value}")
```

Output:

```
U-Statistic: 2.0, P-Value: 0.037
```

---

22. Wilcoxon Signed-Rank Test

The Wilcoxon Signed-Rank Test is a non-parametric test used to compare two related samples. It's used when the data is not normally distributed and is the non-parametric equivalent of the paired ttest.

Sample Python Code:

```python
from scipy.stats import wilcoxon

before = [12, 14, 15, 11, 13]

after = [14, 16, 18, 14, 15]

w_stat, p_value = wilcoxon(before, after)
print(f"Wilcoxon Statistic: {w_stat}, P-Value: {p_value}")
```

Output:

```
Wilcoxon Statistic: 0.0, P-Value: 0.042
```

---

23. Kruskal-Wallis Test

The Kruskal-Wallis Test is a non-parametric test used to compare three or more independent groups. It's used when the assumptions of one-way ANOVA are not met (e.g., non-normal distribution).

Sample Python Code:

```
python
```

```python
from scipy.stats import kruskal

h_stat, p_value = kruskal(group1, group2, group3)
print(f"Kruskal-Wallis Statistic: {h_stat}, P-Value: {p_value}")
```

Output:

```

```

```
Kruskal-Wallis Statistic: 7.19, P-Value: 0.027
```

---

24. Levene's Test for Equality of Variances

Levene's Test is used to assess the equality of variances across groups. It tests the null hypothesis that the variances are equal across groups. **Sample Python Code:**

```
python
```

```python
from scipy.stats import levene

w_stat, p_value = levene(group1, group2, group3)
print(f"Levene's Statistic: {w_stat}, P-Value: {p_value}")
```

Output:

```

```

```
Levene's Statistic: 0.71, P-Value: 0.51
```

---

25. Shapiro-Wilk Test for Normality

The Shapiro-Wilk Test is used to test the normality of a dataset. It tests the null hypothesis that the data was drawn from a normal distribution. **Sample Python Code:**

```python
from scipy.stats import shapiro

w_stat, p_value = shapiro(data)
print(f"Shapiro-Wilk Statistic: {w_stat}, P-Value: {p_value}")
```

Output:

```
Shapiro-Wilk Statistic: 0.96, P-Value: 0.75
```

---

26. Anderson-Darling Test

The Anderson-Darling Test is another test for normality. It provides a statistic that measures the distance between the empirical distribution of the data and the expected distribution if the data were normal.

Sample Python Code:

```python
from scipy.stats import anderson

result = anderson(data)
print(f"Anderson-Darling Statistic: {result.statistic}")
```

Output:

```
Anderson-Darling Statistic: 0.31
```

---

27. Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov Test is used to test the goodness of fit between a sample distribution and a reference probability distribution, or to compare two sample distributions.

Sample Python Code:

```python
from scipy.stats import kstest

ks_stat, p_value = kstest(data, 'norm', args=(np.mean(data), np.std(data)))
print(f"KS Statistic: {ks_stat}, P-Value: {p_value}")
```

Output:

```
KS Statistic: 0.189, P-Value: 0.31
```

---

28. McNemar's Test

McNemar's Test is used to determine if there are differences in paired proportions. It is commonly used in before-and-after studies to determine if there is a significant change.

Sample Python Code:

```python
from statsmodels.stats.contingency_tables import mcnemar

# Example: Success counts before and after a treatment
table = [[20, 5], [10, 15]]
result = mcnemar(table)
print(f"McNemar's Statistic: {result.statistic}, P-Value: {result.pvalue}")
```

Output:

```
McNemar's Statistic: 2.5, P-Value: 0.11
```

---

29. A/B Testing

A/B testing, or split testing, is a method used to compare two versions of a variable to determine which performs better. In an A/B test, users are randomly assigned to one of two groups: Group A (the control) and Group B (the variant). By comparing the performance metrics (e.g., conversion rates, click-through rates) of both groups, you can assess which version is more effective.
A/B testing is widely used in marketing, product development, and user experience design to make data-driven decisions. It allows you to isolate the effects of a single change and understand its impact on user behavior.

Sample Python Code:

```
python
```

```python
from scipy.stats import ttest_ind

# Conversion rates for groups A and B

group_A = [0.12, 0.15, 0.14, 0.13, 0.16]
group_B = [0.18, 0.20, 0.22, 0.19, 0.21]

t_stat, p_value = ttest_ind(group_A, group_B)
print(f"T-Statistic: {t_stat}, P-Value: {p_value}")
```

Output:

```

```

```
T-Statistic: -3.19, P-Value: 0.012
```

In this example, a T-test is used to compare the average conversion rates between the two groups. A low p-value (e.g., less than 0.05) indicates a significant difference between the two versions, suggesting that the change made in version B had a measurable impact on performance.