**PY STAT**

import numpy as np

from scipy import stats

s2 = np.array([86, 47, 45, 47, 40, 97, 98, 75, 65, 83])

1. Descriptive Statistics summaries a dataset, which helps in gaining insights, and making inferences about a dataset.
2. Descriptive Statistics involves estimating centrality measures and measures of dispersion.
3. Mean is the sum of all values divided by a total number of values, of a data set.

print(np.mean(s1))

1. Median is the value that separates the given data set into two halves.

print(np.median(s1))

1. mode function of scipy.stats module can be used for computing mode of a given data set.

print(stats.mode(s1))

##### **Measures of Dispersion**

* Measures of Dispersion provide insights on the spread of given dataset.
* Major measures of dispersion are: range, percentile, inter-quartile range, standard deviation, variance, skewness, and kurtosis.

1. Range is the difference between maximum and minimum values of the dataset.

print(np.ptp(s1))

1. Percentile refers to a value, below which lies given the percentage of data points.

print(np.percentile(s2, 45, interpolation='lower'))

1. Three Quartiles namely, Q1 Q2 and Q3, split the entire dataset into four equal parts.

print(np.percentile(s2, [25, 50, 75], interpolation='lower'))

1. Inter quartile range refers to difference between third quartile (Q3) and first quartile (Q1).

print(stats.iqr(s2, rng=(25, 75), interpolation='lower'))

1. Variance is defined as the average of squared differences, of each data point from dataset's mean.
2. Standard Deviation is square root of variance.

print(np.var(s2))

print(np.std(s2))

1. skewness determines whether the majority of data points are present on one side of the distribution.

A positive value represents right skewed distribution; a negative value represents left skewed one, zero represent unskewed distribution.

print(stats.skew(s2))

1. Kurtosis indicates how much of data is concentrated around mean or shape of the probability distribution.

print(stats.kurtosis(s2))

##### **Random Numbers**

random module of numpy has utilities, which generate arrays of random numbers.

print(np.random.rand())

# generates a 2\*3 array

print(np.random.rand(2,3))

* In statistics, you select items randomly from a population, either with or without a replacement.

print(np.random.choice([11, 22, 33], 2, replace=False))

##### **Random Seeding**

Seed is an important concept when it comes to reproducibility. If you are working with random numbers and you would want to peers to validate your results, i.e., they should also get the same random sequence as you did, you can set the seed to a particular value and send the seed value to your peers.

np.random.seed(100)

print(np.random.rand())

##### **Random Variables**

* In probability theory, the set of all possible outcomes of a random experiment is known as sample space.
* Probabilities of all outcomes of the experiment define the probability distribution.
* A random variable is a variable that takes real numbers or integers and map each value to one of the outcomes of sample space.
* E.g.: In an experiment of tossing a coin, the sample space is {'Head', 'Tail'} and a possible random variable takes the value 0 for head and 1 for the tail.

##### **Probability Distributions**

* There are two types of probability distributions namely discrete and continuous that take integer and real values, respectively.
* scipy.stats module provides classes that represent random variables, corresponding to a large number of probability distributions.
* E.g: the class norm represent normal continuous random variable, and binom represent binomial discrete random variable.

##### **Random Distributions**

scipy.stats module provide a lot of methods for created discrete and continuous random variables.

Commonly used methods are described below.

* pdf / pmf : Probability distribution function (continuous) or probability mass function (discrete).
* cdf : Cumulative distribution function.
* sf : Survival function (1 – cdf).
* rvs : Creating random samples from a distribution.

##### **Random Distributions**

* The following example defines a normal continuous random variable of mean 1.0 and std 2.5.
* It also estimates probabilities and cumulative probabilities at -1, 0 and 1.
* The example also generates six random numbers from defined normal distribution.

#### Example

from scipy import stats

x = stats.norm(loc=1.0, scale=2.5)

print(x.pdf([-1, 0, 1]))

print(x.cdf([-1, 0, 1]))

print(x.rvs((2,3))

Create a normal distribution with mean 32 and standard deviation 4.5.

Set the random seed to 1, and create a random sample of 100 elements from the above defined distribution.

Compute the absolute difference between the sample mean and the distribution mean.

**Import numpy as py**

**from scipy import stats**

**x = stats.norm(loc=32, scale=4.5)**

**np.random.seed(1)**

**print(x.rvs(100))**

#### Problem Statement

Simulate a random experiment of tossing a coin 10000 times, and determine the count of Heads returned.

**Hint:** Define a binomial distribution with n = 1 and p = 0.5.

* Use binom function from scipy.stats.
* Set the random seed to 1.
* Draw a sample of 10000 elements from a defined distribution. Assume that the values '0' and '1' represent Heads and Tails respectively.
* Count the number of 'Heads' and display it. Make used of the 'bincount' method available in 'numpy'.

**Import numpy as py**

**from scipy.stats import binom**

np.random.seed(1)

data=binom.rvs(n=1,p=0.5,size=10000)

print(np.bincount(data))