https://www.pluralsight.com/guides/interpreting-data-using-descriptive-statistics-python

Introduction

Descriptive Statistics is the building block of data science. Advanced analytics is often incomplete without analyzing descriptive statistics of the key metrics. In simple terms, descriptive statistics can be defined as the measures that summarize a given data, and these measures can be broken down further into the measures of central tendency and the measures of dispersion.

Measures of central tendency include mean, median, and the mode, while the measures of variability include standard deviation, variance, and the interquartile range. In this guide, you will learn how to compute these measures of descriptive statistics and use them to interpret the data.

We will cover the topics given below:

1. Mean
2. Median
3. Mode
4. Standard Deviation
5. Variance
6. Interquartile Range
7. Skewness

We will begin by loading the dataset to be used in this guide.

Data

In this guide, we will be using fictitious data of loan applicants containing 600 observations and 10 variables, as described below:

1. Marital_status: Whether the applicant is married ("Yes") or not ("No").
2. Dependents: Number of dependents of the applicant.
3. Is_graduate: Whether the applicant is a graduate ("Yes") or not ("No").
4. Income: Annual Income of the applicant (in USD).
5. Loan_amount: Loan amount (in USD) for which the application was submitted.
6. Term_months: Tenure of the loan (in months).

7. Credit_score: Whether the applicant's credit score was good ("Satisfactory") or not ("Not_satisfactory").
8. Age: The applicant's age in years.
9. Sex: Whether the applicant is female (F) or male (M).
10. approval_status: Whether the loan application was approved ("Yes") or not ("No").

Let's start by loading the required libraries and the data.

python

```
import pandas as pd
import numpy as np
import statistics as st

# Load the data
df =
pd.read_csv("data_desc.csv")
print(df.shape)
print(df.info())
```

Output:

```
(600, 10)

<class
'pandas.core.frame.DataFrame'>
```

```
 RangeIndex: 600 entries, 0 to 599
 Data columns (total 10 columns):
 Marital_status     600 non-null
object
 Dependents         600 non-null int64
 Is_graduate        600 non-null
object
 Income             600 non-null int64
 Loan_amount        600 non-null int64
 Term_months        600 non-null int64
 Credit_score       600 non-null
object
 approval_status    600 non-null
object
 Age                600 non-null int64
 Sex                600 non-null
object
 dtypes: int64(5), object(5)
 memory usage: 47.0+ KB
 None
```

Five of the variables are categorical (labelled as 'object') while the remaining five are numerical (labelled as 'int').

Measures of Central Tendency

Measures of central tendency describe the center of the data, and are often represented by the mean, the median, and the mode.

Mean

Mean represents the arithmetic average of the data. The line of code below prints the mean of the numerical variables in the data. From the output, we can infer that the average age of the applicant is 49 years, the average annual income is USD 705,541, and the average tenure of loans is 183 months. The command df.mean(axis = 0) will also give the same output.

```python
df.mean()
```

Output:

```
Dependents
0.748333
Income
705541.333333
Loan_amount
323793.666667
Term_months       183.350000
Age                49.450000
dtype: float64
```

It is also possible to calculate the mean of a particular variable in a data, as shown below, where we calculate the mean of the variables 'Age' and 'Income'.

```python

print(df.loc[:,'Age'].mean())
print(df.loc[:,'Income'].mean())
```

Output:

```
49.45
705541.33
```

In the previous sections, we computed the column-wise mean. It is also possible to calculate the mean of the rows by specifying the (axis = 1) argument. The code below calculates the mean of the first five rows.

```python
df.mean(axis = 1)[0:5]
```

Output:

```
0     70096.0
1    161274.0
2    125113.4
3    119853.8
4    120653.8
dtype: float64
```

Median

In simple terms, median represents the 50th percentile, or the middle value of the data, that separates the distribution into two halves. The line of code below prints the median of the numerical variables in the data. The command df.median(axis = 0) will also give the same output.

```python
df.median()
```

Output:

```
Dependents          0.0
Income         508350.0
Loan_amount     76000.0
Term_months       192.0
Age                51.0
dtype: float64
```

From the output, we can infer that the median age of the applicants is 51 years, the median annual income is USD 508,350, and the median tenure of loans is 192 months. There is a difference between the mean and the median values of these variables, which is because of the distribution of the data. We will learn more about this in the subsequent sections.

It is also possible to calculate the median of a particular variable in a data, as shown in the *first two lines of code* below. We can also calculate the median of the rows by specifying the (axis = 1) argument. The *third line* below calculates the median of the first five rows.

```python
#to calculate a median of a particular column



print(df.loc[:,'Age'].median())
print(df.loc[:,'Income'].median())

df.median(axis = 1)[0:5]
```

Output:

```
51.0
508350.0

0    102.0
1    192.0
2    192.0
3    192.0
4    192.0
dtype: float64
```

Mode

Mode represents the most frequent value of a variable in the data. This is the only central tendency measure that can be used with categorical variables, unlike the mean and the median which can be used only with quantitative data.

The line of code below prints the mode of all the variables in the data. The .mode() function returns the most common value or most repeated value of a variable. The command df.mode(axis = 0) will also give the same output.

```python
df.mode()
```

Output:

```
   |       | Marital_status
   | Dependents    |
   Is_graduate     | Income
| Loan_amount   | Term_months   |
Credit_score    | approval_status   |
Age     | Sex   |
|---    |---------------   |---------
--- |-------------  |--------   |-----
--------    |-------------  |---------
-----   |-----------------  |-----  |-
----    |
| 0     | Yes               | 0
| Yes           | 333300    | 70000
| 192.0         | Satisfactory  | Yes
| 55    | M     |
```

The interpretation of the mode is simple. The output above shows that most of the applicants are married, as depicted by the 'Marital_status' value of "Yes". Similar interpreation could be done for the other categorical variables like 'Sex' and 'Credit-Score'. For numerical variables, the mode value represents the value that occurs most frequently. For example, the mode value of 55 for the variable 'Age' means that the highest number (or frequency) of applicants are 55 years old.

Measures of Dispersion

In the previous sections, we have discussed the various measures of central tendency. However, as we have seen in the data, the values of these measures differ for many variables. This is because of the extent to which a distribution is stretched or squeezed. In statistics, this is

measured by dispersion which is also referred to as variability, scatter, or spread. The most popular measures of dispersion are standard deviation, variance, and the interquartile range.

Standard Deviation

Standard deviation is a measure that is used to quantify the amount of variation of a set of data values from its mean. A low standard deviation for a variable indicates that the data points tend to be close to its mean, and vice versa. The line of code below prints the standard deviation of all the numerical variables in the data.

```python
df.std()
```

Output:

```
Dependents
1.026362
Income
711421.814154
Loan_amount
724293.480782
Term_months        31.933949
Age                14.728511
dtype: float64
```

While interpreting standard deviation values, it is important to understand them in conjunction with the mean. For example, in the above output, the standard deviation of the variable 'Income' is much higher than that of the variable 'Dependents'. However, the unit of these two variables is different and, therefore, comparing the dispersion of these two variables on the basis of standard deviation alone will be incorrect. This needs to be kept in mind.

It is also possible to calculate the standard deviation of a particular variable, as shown in the *first two lines of code* below. The *third line* calculates the standard deviation for the first five rows.

```python
```

```
print(df.loc[:,'Age'].std())



print(df.loc[:,'Income'].std())

#calculate the standard deviation of
the first five rows
df.std(axis = 1)[0:5]
```

Output:

```
14.728511412020659
711421.814154101

0    133651.842584
1    305660.733951
2    244137.726597
3    233466.205060
4    202769.786470
dtype: float64
```

Variance

Variance is another measure of dispersion. It is the square of the standard deviation and the covariance of the random variable with itself. The line of code below prints the variance of all the numerical variables in the dataset. The interpretation of the variance is similar to that of the standard deviation.

python

```
df.var()
```

Output:

```
 Dependents     1.053420e+00
```

```
Income          5.061210e+11
Loan_amount     5.246010e+11
Term_months     1.019777e+03
Age             2.169290e+02
dtype: float64
```

Interquartile Range (IQR)

The Interquartile Range (IQR) is a measure of statistical dispersion, and is calculated as the difference between the upper quartile (75th percentile) and the lower quartile (25th percentile). The IQR is also a very important measure for identifying outliers and could be visualized using a boxplot.

IQR can be calculated using the iqr() function. The *first line of code* below imports the 'iqr' function from the scipy.stats module, while the *second line* prints the IQR for the variable 'Age'.

python

```python
from scipy.stats import iqr
iqr(df['Age'])
```

Output:

```
25.0
```

Skewness

Another useful statistic is skewness, which is the measure of the symmetry, or lack of it, for a real-valued random variable about its mean. The skewness value can be positive, negative, or undefined. In a perfectly symmetrical distribution, the mean, the median, and the mode will all have the same value. However, the variables in our data are not symmetrical, resulting in different values of the central tendency.

We can calculate the skewness of the numerical variables using the skew() function, as shown below.

python

```
print(df.skew())
```

Output:

```
Dependents     1.169632
Income         5.344587
Loan_amount    5.006374
Term_months   -2.471879
Age           -0.055537
dtype: float64
```

The skewness values can be interpreted in the following manner:

- Highly skewed distribution: If the skewness value is less than −1 or greater than +1.
- Moderately skewed distribution: If the skewness value is between −1 and −½ or between +½ and +1.
- Approximately symmetric distribution: If the skewness value is between −½ and +½.

Putting Everything Together

We have learned the measures of central tendency and dispersion, in the previous sections. It is important to analyse these individually, however, because there are certain useful functions in python that can be called upon to find these values. One such important function is the .describe() function that prints the summary statistic of the numerical variables. The line of code below performs this operation on the data.

python

```
df.describe()
```

Output:

```
```

```
    1      |           | Dependents
    2      | Income         |
    3      Loan_amount      |
    4      Term_months      | Age
    5      |
    6      |-------    |------------
    7      |-------------- |----------
    8      ----    |-------------  |--
    9      ----------  |
   10      | count      | 600.000000
| 6.000000e+02  | 6.000000e+02  |
600.000000       | 600.000000    |
| mean       | 0.748333       |
7.055413e+05    | 3.237937e+05  |
183.350000       | 49.450000     |
| std        | 1.026362       |
7.114218e+05    | 7.242935e+05  |
31.933949        | 14.728511     |
| min        | 0.000000       |
3.000000e+04    | 1.090000e+04  |
18.000000        | 22.000000     |
| 25%        | 0.000000       |
3.849750e+05    | 6.100000e+04  |
192.000000       | 36.000000     |
| 50%        | 0.000000       |
5.083500e+05    | 7.600000e+04  |
192.000000       | 51.000000     |
| 75%        | 1.000000       |
7.661000e+05    | 1.302500e+05  |
192.000000       | 61.000000     |
| max        | 6.000000       |
8.444900e+06    | 7.780000e+06  |
252.000000       | 76.000000     |
```

The above output prints the important summary statistics of all the numerical variables like the mean, median (50%), minimum, and maximum values, along with the standard deviation. We can also calculate the IQR using the 25th and 75th percentile values.

However, the 'describe()' function only prints the statistics

for the quantitative or numerical variable. In order to print the similar statistics for all the variables, an additional argument, include='all', needs to be added, as shown in the line of code below.

```python
df.describe(include='all')
```

Output:

```
|          |
Marital_status  |
Dependents  | Is_graduate
| Income        |
Loan_amount     |
Term_months     |
Credit_score    |
approval_status     | Age
| Sex   |
|--------   |--------------
--  |------------   |------
------- |-------------- |--
------------    |----------
--- |-------------- |-----------------
|------------   |-----  |
| count     | 600               |
600.000000  | 600           |
6.000000e+02    | 6.000000e+02  |
600.000000      | 600           | 600
| 600.000000    | 600   |
| unique    | 2                 | NaN
| 2             | NaN           | NaN
| NaN           | 2             | 2
| NaN           | 2     |
| top       | Yes               | NaN
| Yes           | NaN           | NaN
| NaN           | Satisfactory  | Yes
| NaN           | M     |
| freq      | 391               | NaN
| 470           | NaN           | NaN
```

```
| NaN           | 472           | 410
| NaN           | 489   |
| mean       | NaN              |
0.748333    | NaN           |
7.055413e+05    | 3.237937e+05  |
183.350000      | NaN           | NaN
| 49.450000     | NaN   |
| std        | NaN              |
1.026362    | NaN           |
7.114218e+05    | 7.242935e+05  |
31.933949       | NaN           | NaN
| 14.728511     | NaN   |
| min        | NaN              |
0.000000    | NaN           |
3.000000e+04    | 1.090000e+04  |
18.000000       | NaN           | NaN
| 22.000000     | NaN   |
| 25%        | NaN              |
0.000000    | NaN           |
3.849750e+05    | 6.100000e+04  |
192.000000      | NaN           | NaN
| 36.000000     | NaN   |
| 50%        | NaN              |
0.000000    | NaN           |
5.083500e+05    | 7.600000e+04  |
192.000000      | NaN           | NaN
| 51.000000     | NaN   |
| 75%        | NaN              |
1.000000    | NaN           |
7.661000e+05    | 1.302500e+05  |
192.000000      | NaN           | NaN
| 61.000000     | NaN   |
| max        | NaN              |
6.000000    | NaN           |
8.444900e+06    | 7.780000e+06  |
252.000000      | NaN           | NaN
| 76.000000     | NaN   |
```

Now we have the summary statistics for all the variables. For qualitative variables, we will not have the statistics such

as the mean or the median, but we will have statistics like the frequency and the unique label.

Conclusion

In this guide, you have learned about the fundamentals of the most widely used descriptive statistics and their calculations with Python. We covered the following topics in this guide:

1. Mean
2. Median
3. Mode
4. Standard Deviation
5. Variance
6. Interquartile Range
7. Skewness

It is important to understand the usage of these statistics and which one to use, depending on the problem statement and the data. To learn more about data preparation and building machine learning models using Python's 'scikit-learn' library, please refer to the following guides:

1. Scikit Machine Learning
2. Linear, Lasso, and Ridge Regression with scikit-learn
3. Non-Linear Regression Trees with scikit-learn
4. Machine Learning with Neural Networks Using scikit-learn
5. Validating Machine Learning Models with scikit-learn
6. Ensemble Modeling with scikit-learn
7. Preparing Data for Modeling with scikit-learn