BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Introduction to Statistical Methods

Lab Tutorial

Prof Vineet Garg

BITS Pilani Work Integrated Learning Programmes (WILP)
Bangalore Professional Development Center

Vectra Stock Price

Download the data file (**vectra.csv)**. This file captures data for a company called Vectra for its stock price for a number of days over few years. The different attributes are following: Date, Open, High, Low, Last, Close, Total Trade Quantity and Turnover (Lacs). All numerical values can be assumed to be in the normal distribution.

i. Read the file as Pandas dataframe and examine its different fields. How many rows and columns are present?

ii. Plot *Close* attribute for all the records using Matplotlib and label the axes appropriately.

iii. Gain for the Close attribute is defined as:

$$\text{Gain } \% = \frac{\text{Previous Close Price} - \text{Present Close Price}}{\text{Previous Close Price}} * 100$$

Create a column in the data frame for the *Gain* calculating it w.r.t. the previous record value of *Close*. It can be done using the Pandas method as:

data.Close.pct_change (periods=1)

Where, **periods**=1 specifies that % Gain for a day is calculated from the immediate previous day in the dataset. So, for the very first record %Gain cannot be calculated.

iv. Display the complete the data frame and verify the presence and correctness of the *Gain* field.

v. Establish the 95% confidence interval for the *Gain*. It can be done using the scipy stats package as following:

from scipy import stats

stats.norm.interval (confidence interval, mean, standard deviation)

Here mean and standard deviation can be calculated using Pandas methods for a column like

mean = data.Gain.mean ()

standard deviation = data.Gain.std ()

vi. Calculate the probability that gain will be ≤ 2% on any randomly selected date. It can be done using scipy stats package as following:

stats.norm.cdf (x, mean, standard deviation)

OTT Shows Mean Production Cost

Download the data file (**prodcost.csv)**. This file captures the production cost in lacs of ruppes for 40 OTT shows. It can be assumed that production cost for the population is normally distributed. A researcher claims that the mean production cost now a days is more than 450 lacs. Test the hypothesis taking the significance value as 5%.

i. Form the Hypotheses statements:

H_0: $\mu \leq 450$ and **H_A: $\mu > 450$**

ii. Since, population standard deviation is not known, the hypothesis testing would involve t-distribution. For the significance value = 0.05, the t-critical value can be calculated using the following scipy stats method:

```
from scipy.stats import t

tCrit = t.ppf (1-sigVal, DoF)
```

Note that **t.ppf ()** (percent point function) method calculates the inverse of cumulative distribution function or the quantile of the given probability. The first argument has to be the area on the left hand side for the t-critical value. The second argument is the degree of freedom. So, significance value needs to be subtracted from 1 to get the area on the left hand side to find out the t-critical value.

iii. Perform the 1 sample t-test. The following scipy stats method can be used to perform the one-sample t-test:

```
from scipy import stats

stats.ttest_1samp(a, popmean, alternative='greater')
```

Where, **a** is the sample observations, **popmean** is the expected mean value of the population which is being used for the hypothesis testing and **alternative** specifies the direction of test for alternative hypothesis. Its possible values are: ‘two-sided’, ‘less’ and ‘greater’.

The above ttest method returns t-statistic and p-value respectively that can be compared with t-critical and significance value respectively to test the hypothesis.

Side Note:

From MS-Excel: the sample mean = 429.55 lacks, the standard deviation = 195.0337.

$$\text{t} - \text{statistic} = \frac{429.55 - 450}{\frac{195.0337}{\sqrt{40}}} = -0.6631$$

The calculation can be verified from the Python code as well.

Variance in Parts Size

Download the data file (**parts.csv**). This file captures the diameters in cm of a mechanical part which is circular in shape. The file contains data for a sample of 41 such parts. Factory supervisor claims that the variance in diameter now a days is more than 5. It is resulting from the old machines that demand replacement or heavy repair. Test the claim taking the significance value as 5%.

i. Form the Hypotheses statements:

H_0: $\sigma^2 \leq 5$ and **H_A: $\sigma^2 > 5$**

ii. Since the problem statement talks about the variance, the hypothesis testing would involve chi-squared distribution. For the significance value = 0.05, the chi-squared critical value can be calculated using the following scipy stats method:

chiSqCrit = stats.chi2.ppf (1-(sigVal), DoF)

Note that **chi2.ppf ()** (percent point function) method calculates the inverse of cumulative distribution function or the quantile of the given probability. The first argument has to be the area on the left hand side for the chi-squared critical value. The second argument is the degree of freedom. So, significance value needs to be subtracted from 1 to get the area on the left hand side to find out the chi-squared critical value.

Based on the right-end, left-end or two-tail test requirements, the first argument can be appropriately provided to get the chi-squared critical value(s).

iii. For a right-end tail test:

chiSqCrit = stats.chi2.ppf ((1-sigVal), DoF)

iv. For a left-end tail test:

chiSqCrit = stats.chi2.ppf (sigVal, DoF)

v. For a two tail test:

chiSqCrit1 = stats.chi2.ppf (sigVal/2, DoF)

chiSqCrit2 = stats.chi2.ppf (1-(sigVal/2), DoF)

vi. Chi-squared statistics can be calculated using the formula:

$$\chi^2 = \frac{(n-1)s^2}{\sigma^2}$$

Where n is the sample size, σ^2 is the population variance and s^2 is the sample variance.

vii. The sample variance can be calculated as follows using the observations: **np.var (a, ddof = 1).** Note that ddof = 1 for sample and ddof = 0 for population.

viii. Perform the Hypothesis Testing comparing the chi-squared statistic(s) and critical value.

Plot Continuous Distributions

1. Using a suitable NumPy method, draw 1000 values randomly which are in standard normal distribution and plot them for their PDF.

To yield the reproducible results from any random generator a fixed seed value needs to be used. It can be done in the following way:

np.random.seed (12)

The required random values can be generated using the following method:

x = np.random.randn (1000)

Note that the required shape can be provided as an argument and they need to be sorted before plotting. Above it is one-dimension vector.

The f(x) or y values (PDF) for the generated random values can be calculated using the standard normal distribution function:

$$y = f(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}}$$

Using the matplotlib, the x and y values can be plotted.

import matplotlib.pyplot as plt

plt.plot(x, y)

2. Using a suitable NumPy method, draw 1000 values randomly which are in Gamma distribution and plot them for their PDF. Shape and scale value can be selected as required.

To yield the reproducible results from any random generator a fixed seed value needs to be used. It can be done in the following way:

np.random.seed (12)

The required random values can be generated using the following method:

x = np.random.gamma (alpha, beta, 1000)

Note that alpha is the shape and beta is the scale parameter. The required shape can also be provided as an argument. The generated x values need to be sorted before plotting. Above values are generated it is one-dimension vector.

The f(x) or y values (PDF) for the generated random values can be calculated using the Gamma distribution function:

$$y = f(x) = \begin{cases} \frac{1}{\beta^{\alpha}.\Gamma(\alpha)} x^{\alpha-1} e^{\frac{-x}{\beta}} & \text{for } x \geq 0, \alpha, \beta > 0 \\ 0 & \text{otherwise} \end{cases}$$

Where Gamma function can be calculated using the math package.

import math
math.gamma (alpha)

Using the matplotlib, the x and y values can be plotted.

import matplotlib.pyplot as plt
plt.plot(x, y)

Plot Discrete Probabilities

1. Selecting a suitable sample size (n) and success probability for a desired result (p), plot Binomial probabilities as bar plot.

From 0 to n, linearly and equally spaced points can be generated using the following NumPy method:

```
x = np.linspace (0, n, n+1)
```

From the binom package of scipy the following method can be used generate Binomial Probability Mass Function (PMF):

```
from scipy.stats import binom

pmf = binom.pmf (x, n, p)
```

Using the following matplotlib method the bar plot can be drawn for the PMF values:

```
import matplotlib.pyplot as plt

plt.xlim (-1, n+1)

plt.bar (x, pmf, color='b')
```

2. Selecting a suitable long run average (λ), plot Poisson probabilities as bar plot.

From 0 to n, linearly and equally spaced points can be generated using the following NumPy method:

```
x = np.linspace (0, n, n+1)
```

From the poisson package of scipy the following method can be used generate Poisson Probability Mass Function (PMF):

```
from scipy.stats import poisson

pmf = poisson.pmf (x, lam)
```

Using the following matplotlib method the bar plot can be drawn for the PMF values:

```
import matplotlib.pyplot as plt

plt.xlim (-1, n+1)

plt.bar (x, pmf, color='r')
```

Draw Random Points with Gaussian Noise

Many times during the Machine Learning simulation, random data points are required added with some controlled Gaussian Noise. This lab exercise is attempted to demonstrate the requirement.

i. Generate 100 random points in uniform distribution which range from [-3.0, +3.0). This can be achieved using the following NumPy method:

```
np.random.seed (12)
m = 100
x = 6 * np.random.rand (m, 1) – 3
```

Note that np.random.rand () method generates Uniformly distributed random numbers in the range of [0.0, 1.0). So multiplying the range with 6 and subtracting the 3 from them will yield a range of [-3.0, +3.0).

ii. Now, any suitable function can be selected to generate y or f(x) values for these x values. For example:

$$y = f(x) = 0.5x^2 + x + 2$$

iii. The Gaussian noise can be generated and added to y or f(x) it using the following NumPy method:

```
np.random.randn (m, 1)
```

Note that the above method is just generating random values in standard normal distribution (Gaussian).

iv. Using the following matplotlib method, the y values can be plotted:

```
import matplotlib.pyplot as plt
plt.plot (x,y, '.', markersize = 8)
```

Thank You