RAMAIAH
Institute of
Technology

Department of Information Science & Engineering

BE COURSE PROJECT

ISEE1 Data Science

Normal Distribution

Submitted By: **1MS16IS003** **Abhishek R Mishra**
1MS16IS007 **Amogh Simha**

Faculty In-Charge: Dr. Krishnaraj P M

RAMAIAH INSTITUTE OF TECHNOLOGY
(AUTONOMOUS INSTITUTE, AFFILIATED TO VTU)
Bangalore – 560054
2019-2020

1. Introduction

A **Normal Distribution** is a continuous probability distribution. The normal distribution is sometimes referred to as a bell curve. That means that we expect the value to be 0 (on average) but the actual realized values of our random variable 'wiggle' around 0 (Mean). The amount that it wiggles by is 1 (SD). The normal distribution is defined by the following probability density function -

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

The normal distribution model is motivated by the **Central Limit Theorem**. This theory states that averages calculated from independent, identically distributed random variables have approximately normal distributions, regardless of the type of distribution from which the variables are sampled.

1.1 Properties

Some of the properties of a standard normal distribution are mentioned below:

- The normal curve is symmetric about the mean and bell shaped.
- Mean, mode and median is zero which is the centre of the curve.
- Approximately 68% of the data will be between -1 and +1 (i.e. within 1 standard deviation from the mean), 95% between -2 and +2 (within 2 SD from the mean) and 99.7% between -3 and 3 (within 3 SD from the mean)
- Normal distribution is symmetrical distribution, but not all symmetrical distributions are normal.
- Normal distribution is the proper term for a probability bell curve.

2. Libraries

1. NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

1.1 np.random.normal

Draw random samples from a normal (Gaussian) distribution. The probability density function of the normal distribution, first derived by De Moivre and 200 years later by both Gauss and Laplace independently, is often called the bell curve because of its characteristic shape.

Parameters-
Loc: Mean ("centre") of the distribution.
Scale: Standard deviation (spread or "width") of the distribution.
Size: Output shape.

2. Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code.

2.1 matplotlib.pyplot

matplotlib.pyplot is a state-based interface to matplotlib. It provides a MATLAB-like way of plotting. pyplot is mainly intended for interactive plots and simple cases of programmatic plot generation.

plot(*args, scalex, scaley, data) – Plot y v/s x as lines or markers.

3. Pandas

Pandas is an open source library providing high performance, easy-to-use data structures and data analysis tools. Provides tools for **reading and writing data** between in-memory data structures and different formats like CSV and text files, Microsoft Excel, SQL databases etc.

read_csv('filename.extension') – Read a dataset of csv, text and save it in a variable.

4. sklearn.preprocessing

The package provides several common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for the downstream estimators.

4.1 StandardScaler

Standardize features by removing the mean and scaling to unit variance. It is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data (e.g. Gaussian with 0 mean and unit variance).

fit_transform(self, X[, y]) – Fit to data then transform

Data Science Assignment - Normal Distribution

Abhishek Mishra 1MS16IS003
Amogh Simha 1MS16IS007

A Normal Distribution is a continuous probability distribution. It is sometimes referred to as a "bell curve". A normally distributed random variable might have a mean of 0 and a standard deviation of 1.

Why is it Useful?

Many real world phenomena conform to the normal distribution.
For example, people's heights are famously normally distributed. This is applicable for a large sample of data.
Statistical inference and hypothesis testing relies heavily on the normal distribution.
Return of assets like stock are assumed to follow normal distrbution.

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

$$\begin{aligned}
\mu &= \text{ Mean} \\
\sigma &= \text{ Standard Deviation} \\
\pi &\approx 3.14159\cdots \\
e &\approx 2.71828\cdots
\end{aligned}$$

Method 1: Using Randomized Values

Import Libraries

Importing numpy to generate the numbers
matplotlib is used for plotting the data

In [12]:
```
import numpy as np
import matplotlib.pyplot as plt
```

Create a class with methods to calculate and plot the data

Python class that calculates the bell curve to be plotted using the normal distribution formula

In [13]:
```
class norm1:
    def __init__(self, a1, b1, c1):
        self.a1 = a1
        self.b1 = b1
        self.c1 = c1

    def dist_curve(self):
        plt.plot(self.c1, 1/(self.b1 * np.sqrt(2 * np.pi)) *
            np.exp( - (self.c1 - self.a1)**2 / (2 * self.b1**2) ), linewidth=2
, color='green')
        plt.show()
```

Set mean=0 and SD=1

In [14]:
```
mean = 0
sd = 1
```

Create 10000 normalized, random values

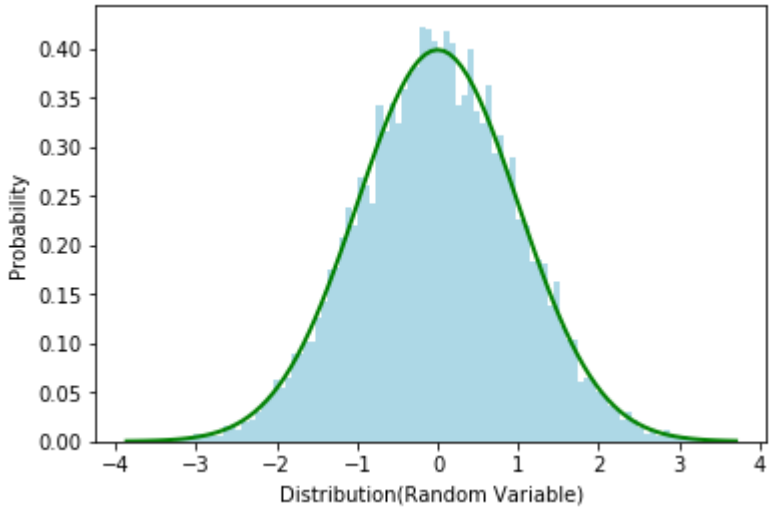
numpy function to generate randomized numbers that are normalized with the required mean and standard deviation

In [15]:
```
c = np.random.normal(mean, sd, 10000)
```

Plot and Visualize the data

Histogram plot of the normalized data along with the bell curve

In [22]:
```
w1, x1, z1 = plt.hist(c, 100, normed=True,color='lightblue')
plt.xlabel("Distribution(Random Variable)")
plt.ylabel("Probability")
hist1 = norm1(mean, sd, x1)
plot1 = hist1.dist_curve()
```



Method 2: Using dataset

Using a dataset containing weight & height data

Import Libraries

In [17]:
```
import pandas as pd
```

Import Dataset

Dataset contains 10000 tuples read using pandas

In [18]:
```
data = pd.read_csv("weight-height.csv")
```

Dataset Configuration

In [19]:
```
data1 = data.drop(["Gender"],axis=1)
print(data1.head())
print(data1.shape)
```

```
      Height      Weight
0  73.847017  241.893563
1  68.781904  162.310473
2  74.110105  212.740856
3  71.730978  220.042470
4  69.881796  206.349801
(10000, 2)
```

Normalize with StandardScaler

StandardScaler is used to transform the dataset to have mean=0 and SD=1

In [20]:
```
from sklearn.preprocessing import StandardScaler
newdata = StandardScaler().fit_transform(data1)
newdata
```

Out[20]:
```
array([[ 1.94406149,  2.50579697],
       [ 0.62753668,  0.02710064],
       [ 2.01244346,  1.59780623],
       ...,
       [-0.64968792, -1.02672965],
       [ 0.69312469,  0.07512745],
       [-1.14970831, -1.48850724]])
```

Plot and Display

Histogram plot of normalized data along with bell curve
Comparison with Non-normal weight plot

In [21]:
```
h=newdata[:,0]
w=newdata[:,1]

w2,x2,z2=plt.hist(h,color=['lightblue'],bins=100,normed=True)
plt.xlabel("Distribution(Height)")
plt.ylabel("Probability")
plt.title("Normal Distribution of Height")
hist2 = norm1(0, 1, x2)
plot2 = hist2.dist_curve()
plt.show()

#plt.hist(w,color=['green'],bins=100)
#plt.xlabel("Weight")
#plt.ylabel("Probability")
#plt.title("Weight Distribution")
#plt.show()
```

```
C:\Users\Administrator\Anaconda\lib\site-packages\matplotlib\axes\_axes.py:6521: MatplotlibDeprecationWarning:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in 3.1. Use 'density' instead.
  alternative="'density'", removal="3.1")
```