Assignment_2

July 20, 2017

1 How to Normalize & Standardize Randomly Generated Data

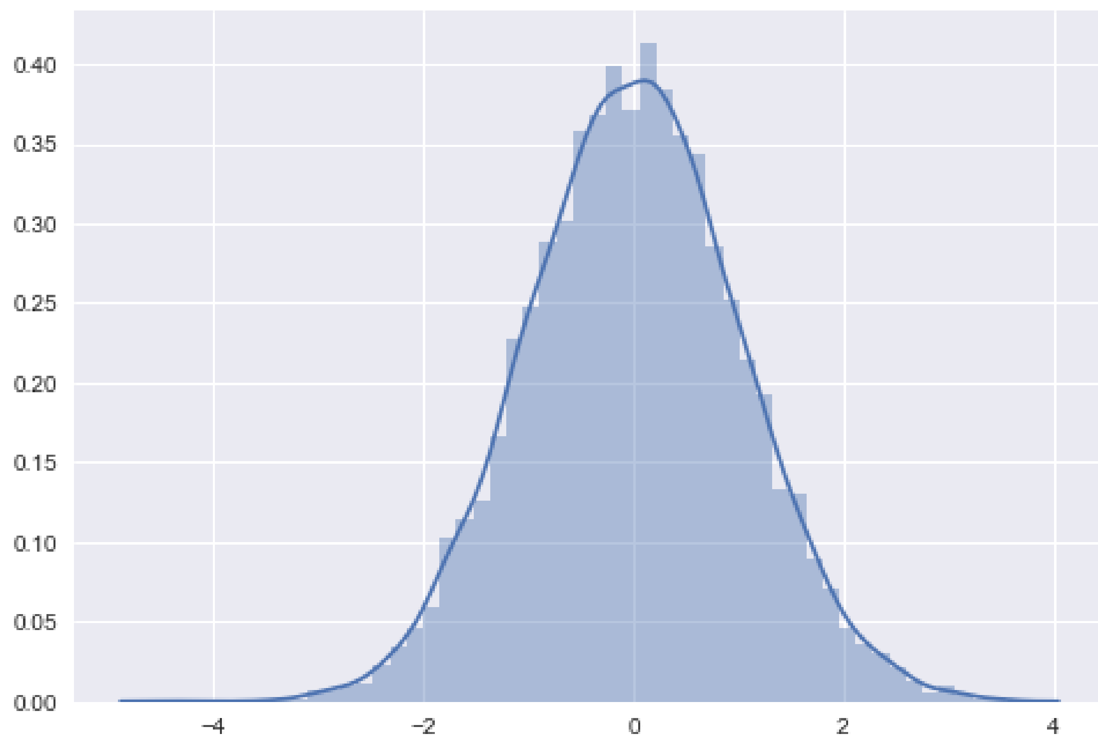
Some machine learning algorithms will achieve better performance if your data has a consistent scale or distribution.

Two techniques that you can use to consistently rescale your data are normalization and standardization.

```
In [1]: # Lets get the 1-D data using numpy as follows:
        import numpy as np
        def genarate_data():
            return np.random.randn(10000)

        data = genarate_data()
```

1.1 Analysing the Distribution of Random Generated Data

```
In [2]: # Now we will check if the distribution is normal or not.
        import seaborn as sns
        import matplotlib.pyplot as plt

        sns.distplot(data)
        plt.show()
```




```
In [3]: # It is clear that the data we have, is having the Gausian Distribution.
        # Now lets try to normalize (N) and standardize (S) the data and /
        #plot (pdf plots) of the same.
```

1.2 Normalizing the data

Normalization is a rescaling of the data from the original range so that all values are within the range of 0 and 1.

A value is normalized as follows:

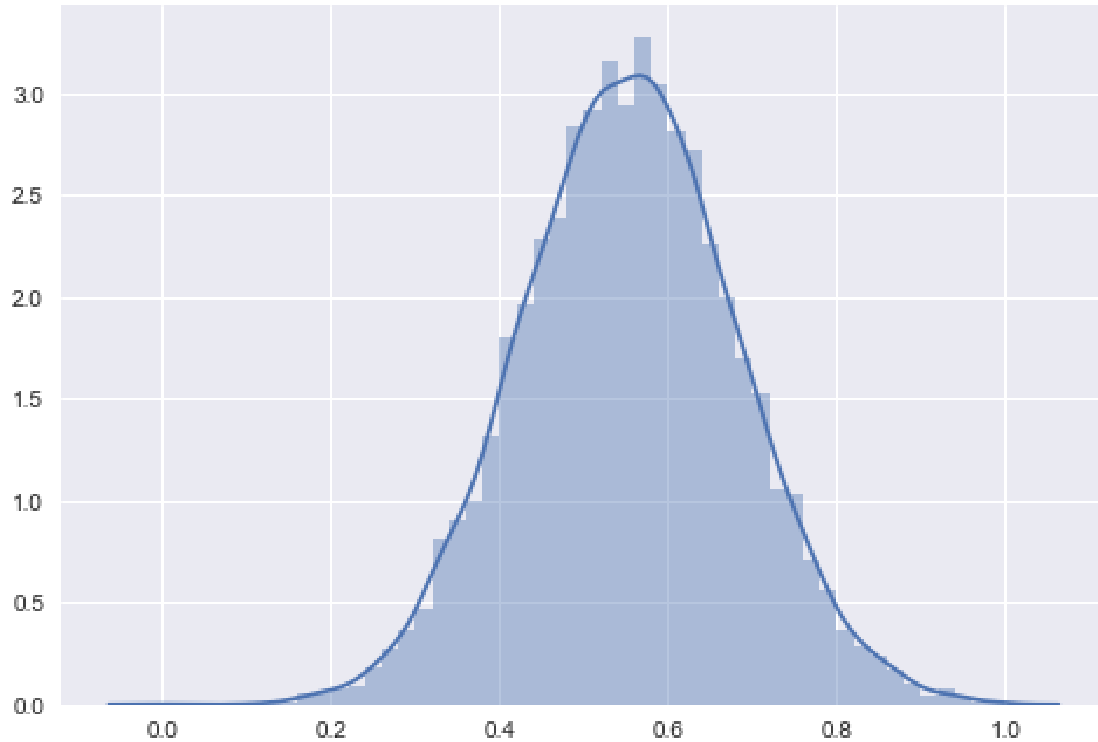
y = (x - min) / (max - min)

Where the minimum and maximum values pertain to the value x being normalized.

You can see that if an 'x' value is provided that is outside the bounds of the minimum and maximum values, then the resulting value will not be in the range of 0 and 1. You could check for these observations prior to making predictions and either remove them from the dataset or limit them to the pre-defined maximum or minimum values.

```
In [4]: # First we will normalize the data.
        # A value is normalized as follows:
        # value = (value - min)/(max - min)
        denominator = max(data) - min(data)
        normalize = []
        for i in range(0,len(data)):
            normalize.append((data[i] - min(data))/denominator)
```

```
sns.distplot(normalize)
plt.show()
```



1.3 Sandardizing the Data

Standardizing a dataset involves rescaling the distribution of values so that the mean of observed values is 0 and the standard deviation is 1. This can be thought of as subtracting the mean value or centering the data.

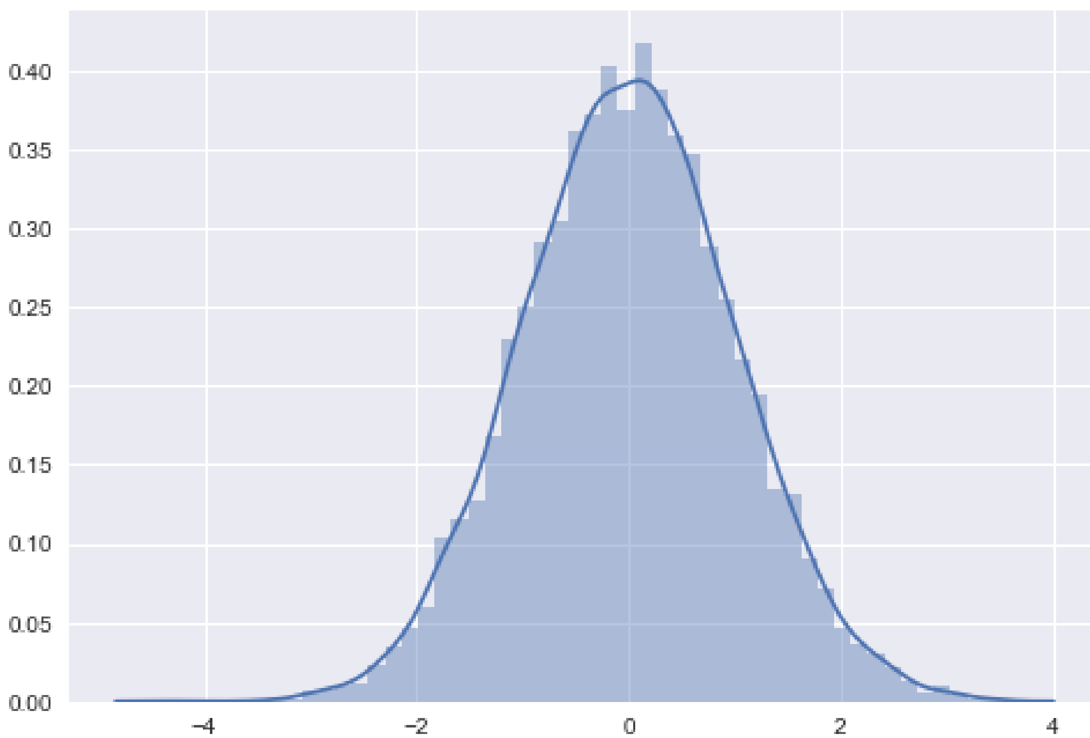
Standardization assumes that your observations fit a Gaussian distribution (bell curve) with a well behaved mean and standard deviation. You can still standardize your data if this expectation is not met, but you may not get reliable results.

A value is standardized as follows: y = (x - mean) / standard_deviation

Like normalization, standardization can be useful, and even required in some machine learning algorithms when your data has input values with differing scales.

```
In [5]: # Now we will standardize the data.
        # A value is standardized as follows:
        # value = (value - mean) / standard_deviation

        def variance(data):
            m = sum(data)/float(len(data))
            v=0
```

```
    for i in data:
        v+=(i - m)**2
    return v/float(len(data))

def std_dev(variance):
    return variance**0.5

m = sum(data)/float(len(data))
var = variance(data)
sd = std_dev(var)
standardize = []
for i in range(0, len(data)):
    standardize.append((data[i] - m)/sd)

sns.distplot(standardize)
plt.show()
```



1.4 Expectation of Normalized and Standardized Data

Expectation of a normal distribution is the mean of that data.

```
In [6]: print "Expectation of Normalized data is %f" % (sum(normalize)/len(normalize))

        print "Expectation of Standardized data is %f" % (sum(standardize)/len(standardize))
```

```
Expectation of Normalized data is 0.553000
Expectation of Standardized data is 0.000000
```