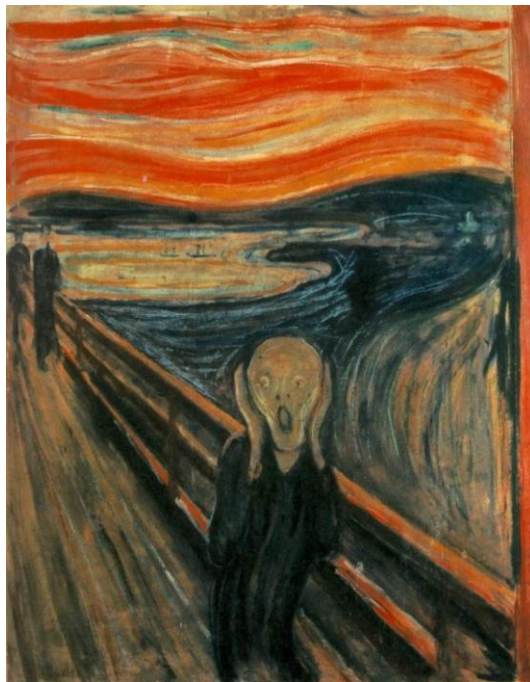


DR. ALVIN'S PUBLICATIONS

# NORMALIZER, SCALER, BUCKETIZER AND BINARIZER

---

USING PYSPARK  
DR. ALVIN ANG



---

1 | PAGE

COPYRIGHTED BY DR ALVIN ANG  
WWW.ALVINANG.SG

# CONTENTS

<b>I. Start a Spark Session .....</b>	<b>3</b>
<b>A. Import the Data .....</b>	<b>4</b>
<b>II. Vector Assembler .....</b>	<b>6</b>
<b>III. Normalizer.....</b>	<b>7</b>
<b>IV. Binerizer .....</b>	<b>9</b>
<b>V. Standard Scalar.....</b>	<b>10</b>
<b>VI. Bucketizer.....</b>	<b>12</b>
<b>About Dr. Alvin Ang .....</b>	<b>15</b>

---

## I. START A SPARK SESSION

---

- This article presents common transformations required for data preprocessing and feature engineering.
- These help in preparing data the right way for applying machine learning.

Most of the stuff here are abstracted from:

<https://www.amazon.com/Learn-PySpark-Python-based-Machine-Learning/dp/1484249607>

IPYNB:

[https://www.alvinang.sg/s/Normalizer\\_Scaler\\_Bucketizer\\_and\\_Binarizer\\_with\\_PySpark\\_by\\_Dr\\_Alvin\\_Ang.ipynb](https://www.alvinang.sg/s/Normalizer_Scaler_Bucketizer_and_Binarizer_with_PySpark_by_Dr_Alvin_Ang.ipynb)

File Used: <https://www.alvinang.sg/s/transformations.csv>

First, you need to install PySpark into Google Colab.

Follow the steps here:

- <https://tatwan.github.io/blog/colab/python/spark/2020/01/06/Colab-Spark-Instructions.html>

Or....

```
[4] !apt-get install openjdk-8-jdk-headless -qq > /dev/null
[5] !wget -q https://dlcdn.apache.org/spark/spark-3.2.1/spark-3.2.1-bin-hadoop3.2.tgz
[6] !tar xf spark-3.2.1-bin-hadoop3.2.tgz
[7] !pip install -q findspark
[8] import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.2.1-bin-hadoop3.2"
[9] os.environ["SPARK_HOME"]
'/content/spark-3.2.1-bin-hadoop3.2'
[10] import findspark
findspark.init()
[11] from pyspark.sql import SparkSession
spark = SparkSession.builder.master("local[*]").getOrCreate()
[12] print(spark.version)
3.2.1
```

#### A. IMPORT THE DATA

```
Importing the Transform.csv
[10] from pyspark import SparkFiles
url = 'https://www.alvinang.sg/s/transformations.csv'
spark.sparkContext.addFile(url)
df = spark.read.csv(SparkFiles.get("transformations.csv"), header=True, inferSchema=True)
df.count()
20
```

```
[12] df.show()
```

Col 1	Col 2	Col 3	Col 4	Col 5	Col 6	Col 7	Col 8	label
3	32	9.0	3	3	17	2	5	0.1111111
3	27	13.0	3	1	14	3	4	3.2307692
4	22	2.5	0	1	16	3	5	1.3999996
4	37	16.5	4	3	16	5	5	0.7272727
5	27	9.0	1	1	14	3	4	4.666666
4	27	9.0	0	2	14	3	4	4.666666
5	37	23.0	6	2	12	5	4	0.8521735
5	37	23.0	6	2	12	2	3	1.826086
3	22	2.5	0	2	12	3	3	4.7999992
3	27	6.0	0	1	16	3	5	1.3333333
2	27	6.0	2	1	16	3	5	3.2666645
5	27	6.0	2	3	14	3	5	2.041666
3	37	16.5	6	1	12	2	3	0.4848484
5	27	6.0	0	2	14	3	2	2.0
4	22	6.0	1	1	14	4	4	3.2666645
4	37	9.0	2	2	14	3	6	1.3611107
4	27	6.0	1	1	12	3	5	2.0
1	37	23.0	6	4	14	5	2	1.826086
2	42	23.0	2	2	20	4	4	1.826086
4	37	6.0	0	2	16	5	4	2.041666

## II. VECTOR ASSEMBLER

### Vector Assembler

```
from pyspark.ml.feature import VectorAssembler

assembler = VectorAssembler(inputCols=[col for col in df.columns if col != 'label'], \
                             outputCol="features")
```

```
[14] df_new=assembler.transform(df)
```

```
[20] df_new.show()
```

Col 1	Col 2	Col 3	Col 4	Col 5	Col 6	Col 7	Col 8	label	features
3	32	9.0	3	3	17	2	5	0.11111111	[3.0,32.0,9.0,3.0...
3	27	13.0	3	1	14	3	4	3.2307692	[3.0,27.0,13.0,3.0...
4	22	2.5	0	1	16	3	5	1.39999996	[4.0,22.0,2.5,0.0...
4	37	16.5	4	3	16	5	5	0.7272727	[4.0,37.0,16.5,4.0...
5	27	9.0	1	1	14	3	4	4.6666666	[5.0,27.0,9.0,1.0...
4	27	9.0	0	2	14	3	4	4.6666666	[4.0,27.0,9.0,0.0...
5	37	23.0	6	2	12	5	4	0.8521735	[5.0,37.0,23.0,6.0...
5	37	23.0	6	2	12	2	3	1.826086	[5.0,37.0,23.0,6.0...
3	22	2.5	0	2	12	3	3	4.7999992	[3.0,22.0,2.5,0.0...
3	27	6.0	0	1	16	3	5	1.3333333	[3.0,27.0,6.0,0.0...
2	27	6.0	2	1	16	3	5	3.2666645	[2.0,27.0,6.0,2.0...
5	27	6.0	2	3	14	3	5	2.041666	[5.0,27.0,6.0,2.0...
3	37	16.5	6	1	12	2	3	0.4848484	[3.0,37.0,16.5,6.0...
5	27	6.0	0	2	14	3	2	2.0	[5.0,27.0,6.0,0.0...
4	22	6.0	1	1	14	4	4	3.2666645	[4.0,22.0,6.0,1.0...
4	37	9.0	2	2	14	3	6	1.3611107	[4.0,37.0,9.0,2.0...
4	27	6.0	1	1	12	3	5	2.0	[4.0,27.0,6.0,1.0...
1	37	23.0	6	4	14	5	2	1.826086	[1.0,37.0,23.0,6.0...
2	42	23.0	2	2	20	4	4	1.826086	[2.0,42.0,23.0,2.0...
4	37	6.0	0	2	16	5	4	2.041666	[4.0,37.0,6.0,0.0...

the vector assembler took the first 8 columns and placed it into 'features' column

### III. NORMALIZER

## NORMALIZER

```
[21] from pyspark.ml.feature import Normalizer
```

```
[22] normalizer = Normalizer(inputCol="features", outputCol="norm_features", p=1.0)
```

▶ normalizer

↳ Normalizer\_fa7d8dd904d0

- Normalization refers to transformation of data in such a way that the new normalized data has a mean of 0 and a standard deviation of 1.

```
[24] normalised_l1_data = normalizer.transform(df_new)
```

```
[26] normalised_l1_data.show()
```

Col 1	Col 2	Col 3	Col 4	Col 5	Col 6	Col 7	Col 8	label	feature_1	norm_features
3	32	9.0	3	3	17	2	5	0.11111111	[3.0,32.0,9.0,3.0...	[0.04054054054054...
3	27	13.0	3	1	14	3	4	3.2307692	[3.0,27.0,13.0,3...	[0.04411764705882...
4	22	2.5	0	1	16	3	5	1.39999996	[4.0,22.0,2.5,0.0...	[0.07476635514018...
4	37	16.5	4	3	16	5	5	0.7272727	[4.0,37.0,16.5,4...	[0.04419889502762...
5	27	9.0	1	1	14	3	4	4.6666666	[5.0,27.0,9.0,1.0...	[0.078125,0.42187...
4	27	9.0	0	2	14	3	4	4.6666666	[4.0,27.0,9.0,0.0...	[0.06349206349206...
5	37	23.0	6	2	12	5	4	0.8521735	[5.0,37.0,23.0,6...	[0.05319148936170...
5	37	23.0	6	2	12	2	3	1.826086	[5.0,37.0,23.0,6...	[0.05555555555555...
3	22	2.5	0	2	12	3	3	4.7999992	[3.0,22.0,2.5,0.0...	[0.06315789473684...
3	27	6.0	0	1	16	3	5	1.3333333	[3.0,27.0,6.0,0.0...	[0.04918032786885...
2	27	6.0	2	1	16	3	5	3.2666645	[2.0,27.0,6.0,2.0...	[0.03225806451612...
5	27	6.0	2	3	14	3	5	2.041666	[5.0,27.0,6.0,2.0...	[0.07692307692307...
3	37	16.5	6	1	12	2	3	0.4848484	[3.0,37.0,16.5,6...	[0.03726708074534...
5	27	6.0	0	2	14	3	2	2.0	[5.0,27.0,6.0,0.0...	[0.08474576271186...
4	22	6.0	1	1	14	4	4	3.2666645	[4.0,22.0,6.0,1.0...	[0.07142857142857...
4	37	9.0	2	2	14	3	6	1.3611107	[4.0,37.0,9.0,2.0...	[0.05194805194805...
4	27	6.0	1	1	12	3	5	2.0	[4.0,27.0,6.0,1.0...	[0.06779661016949...
1	37	23.0	6	4	14	5	2	1.826086	[1.0,37.0,23.0,6...	[0.01086956521739...
2	42	23.0	2	2	20	4	4	1.826086	[2.0,42.0,23.0,2...	[0.02020202020202...
4	37	6.0	0	2	16	5	4	2.041666	[4.0,37.0,6.0,0.0...	[0.05405405405405...

norm\_features took the 'features' and normalized it....

```

normalised_l1_data.select('norm_features').show(truncate=False)
+-----+-----+-----+-----+-----+-----+-----+-----+
| norm_features | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
+-----+-----+-----+-----+-----+-----+-----+-----+
| [0.04054054054054054, 0.4324324324324324, 0.1216216216216216, 0.04054054054054054, 0.04054054054054054, 0.2297297297297297, 0.02702702702702703, 0.06756756756756757] |
| [0.04411764705882353, 0.39705882352941174, 0.19117647058823528, 0.04411764705882353, 0.01470588235294117, 0.2058823529411764, 0.0441176470588235, 0.058823529411764705] |
| [0.07476635514018691, 0.411214953271028, 0.1672897196261682, 0.0, 0.018691588785046, 0.28, 0.29906542056074, 0.64, 0.05607476635514, 0.186, 0.0934579439252, 0.364] |
| [0.04419889502762431, 0.4088397790055249, 0.8232044198895028, 0.04419889502762431, 0.03314917127071823, 0.17679558011049723, 0.055248618784530384, 0.055248618784530384] |
| [0.078125, 0.421875, 0.0625, 0.015625, 0.015625, 0.21875, 0.046875, 0.0625] |
| [0.06349206349206349, 0.4285714285714285, 0.14285714285714285, 0.0, 0.0317460317460, 0.1744, 0.222222222222, 0.222, 0.0476190476190, 0.7616, 0.063492063492, 0.6349] |
| [0.05319148936170213, 0.39361702127659576, 0.2468885106382978, 0.06382978723404255, 0.02127659574468085, 0.127659574468085, 0.05319148936170213, 0.0425531914893617] |
| [0.05555555555555555, 0.4111111111111111, 0.5555555555555554, 0.06666666666666667, 0.02222222222222223, 0.13333333333333333, 0.02222222222222222, 0.03333333333333333] |
| [0.06315789473684211, 0.4631578947368421, 0.5263157894736842, 0.0, 0.04210526315789, 0.736, 0.2526315789473, 0.843, 0.0631578947368, 0.211, 0.0631578947368, 0.211] |
| [0.04918032786885246, 0.4426229508196721, 0.9836065573770492, 0.0, 0.01639344262295, 0.82, 0.26229508196721, 0.13, 0.04918032786885, 0.46, 0.08196721311475, 0.09] |
| [0.03225806451612903, 0.43548387096774194, 0.0967741935483871, 0.03225806451612903, 0.016129032258064516, 0.25806451612903225, 0.04838709677419355, 0.08064516129032258] |
| [0.07692307692307693, 0.4153846153846154, 0.9230769230769231, 0.03076923076923077, 0.046153846153846156, 0.2153846153846154, 0.046153846153846156, 0.07692307692307693] |
| [0.037267080745341616, 0.45962732919254656, 0.2049689440993788, 0.0745341614906832, 0.0124223602484472, 0.149068322981366, 0.024844720496894, 0.08, 0.037267080745341616] |
| [0.0847457627118644, 0.4576271186440678, 0.16949152542373, 0.0, 0.0338983050847457, 0.2372881355932203, 0.0508474576271186, 0.0338983050847457] |
| [0.07142857142857142, 0.39285714285714285, 0.10714285714285714, 0.01785714285714285, 0.0178571428571428, 0.0178571428571428, 0.025, 0.0714285714, 0.857142, 0.0714285714, 0.857142] |
| [0.05194805194805195, 0.4805194805194805, 0.1688311688311688, 0.025974025974025976, 0.02597402597402597, 0.1818181818181818, 0.0389610389610389, 0.07792207792207792] |
| [0.06779661016949153, 0.4576271186440678, 0.016949152542373, 0.01694915254237288, 0.01694915254237288, 0.2033898305084746, 0.5084745762711865, 0.847457627118644] |
| [0.010869565217391304, 0.40217391304347827, 0.25, 0.06521739130, 0.34782, 0.04347826086, 0.565216, 0.1521739130, 0.347827, 0.0543478260, 0.695652, 0.0217391304, 0.4782608] |
| [0.020202020202020204, 0.42424242424242425, 0.2323232323232323, 0.020202020202020204, 0.020202020202020204, 0.020202020202020204, 0.020202020202020204, 0.04040404040404041, 0.04040404040404041] |
| [0.05405405405405406, 0.5, 0.08108108108108109, 0.0, 0.0, 0.02702702, 0.2702703, 0.21621621, 0.1621623, 0.067567567, 0.6756757, 0.054054054, 0.5405406] |

```

- As can be seen, the 'norm-features' column consists of 8 columns
- These 8 columns show that the previous 8 columns has been "normalized" to fit into values between 0 and 1.



---

#### IV. BINERIZER

---

```
from pyspark.ml.feature import Binarizer

binarizer = Binarizer(threshold=0.99, inputCol="label",\
                      outputCol="binarized_label")
```

```
binarizer
```

```
Binarizer_7aaaccf1cb0b
```

```
new_df=binarizer.transform(df)
```

```
new_df.show()
```

Col 1	Col 2	Col 3	Col 4	Col 5	Col 6	Col 7	Col 8	label	binarized_label
3	32	9.0	3	3	17	2	5	0.1111111	0.0
3	27	13.0	3	1	14	3	4	3.2307692	1.0
4	22	2.5	0	1	16	3	5	1.3999996	1.0
4	37	16.5	4	3	16	5	5	0.7272727	0.0
5	27	9.0	1	1	14	3	4	4.6666666	1.0
4	27	9.0	0	2	14	3	4	4.6666666	1.0
5	37	23.0	6	2	12	5	4	0.8521735	0.0
5	37	23.0	6	2	12	2	3	1.826086	1.0
3	22	2.5	0	2	12	3	3	4.7999992	1.0
3	27	6.0	0	1	16	3	5	1.3333333	1.0
2	27	6.0	2	1	16	3	5	3.2666645	1.0
5	27	6.0	2	3	14	3	5	2.041666	1.0
3	37	16.5	6	1	12	2	3	0.4848484	0.0
5	27	6.0	0	2	14	3	2	2.0	1.0
4	22	6.0	1	1	14	4	4	3.2666645	1.0
4	37	9.0	2	2	14	3	6	1.3611107	1.0
4	27	6.0	1	1	12	3	5	2.0	1.0
1	37	23.0	6	4	14	5	2	1.826086	1.0
2	42	23.0	2	2	20	4	4	1.826086	1.0
4	37	6.0	0	2	16	5	4	2.041666	1.0

0 if label starts with 0

1 if label starts with  
1 or more

- Because threshold is 0.99
  - Anything below 1 is deemed 0
  - Anything above 1 is deemed 1

---

## V. STANDARD SCALAR

---

```
from pyspark.ml.feature import StandardScaler

scaler = StandardScaler (inputCol="features",\
                          outputCol="scaled_features",\
                          withStd=False, withMean=True)
```

```
scaler
```

```
StandardScaler_87e40a39e1cc
```

```
scaler_model = scaler.fit(df_new)
```

```
scaler_model
```

```
StandardScalerModel: uid=StandardScaler_87e40a39e1cc, numFeatures=8, withMean=true, withStd=false
```

- Uses the unbiased sample standard deviation instead of the population standard deviation.
- The “unit std” is computed using the corrected sample standard deviation, which is computed as the square root of the unbiased sample variance.
- In other words, after scaling, don’t expect it to be between 0 and 1.

```
scaled_data = scaler_model.transform(df_new)
```

```
scaled_data.show()
```

Col 1	Col 2	Col 3	Col 4	Col 5	Col 6	Col 7	Col 8	label	features	scaled_features
3	32	9.0	3	3	17	2	5	0.1111111	[3.0,32.0,9.0,3.0...	[-0.65000000000000...
3	27	13.0	3	1	14	3	4	3.2307692	[3.0,27.0,13.0,3...	[-0.65000000000000...
4	22	2.5	0	1	16	3	5	1.39999996	[4.0,22.0,2.5,0.0...	[0.34999999999999...
4	37	16.5	4	3	16	5	5	0.7272727	[4.0,37.0,16.5,4...	[0.34999999999999...
5	27	9.0	1	1	14	3	4	4.6666666	[5.0,27.0,9.0,1.0...	[1.34999999999999...
4	27	9.0	0	2	14	3	4	4.6666666	[4.0,27.0,9.0,0.0...	[0.34999999999999...
5	37	23.0	6	2	12	5	4	0.8521735	[5.0,37.0,23.0,6...	[1.34999999999999...
5	37	23.0	6	2	12	2	3	1.826086	[5.0,37.0,23.0,6...	[1.34999999999999...
3	22	2.5	0	2	12	3	3	4.79999992	[3.0,22.0,2.5,0.0...	[-0.65000000000000...
3	27	6.0	0	1	16	3	5	1.3333333	[3.0,27.0,6.0,0.0...	[-0.65000000000000...
2	27	6.0	2	1	16	3	5	3.2666645	[2.0,27.0,6.0,2.0...	[-1.65000000000000...
5	27	6.0	2	3	14	3	5	2.041666	[5.0,27.0,6.0,2.0...	[1.34999999999999...
3	37	16.5	6	1	12	2	3	0.4848484	[3.0,37.0,16.5,6...	[-0.65000000000000...
5	27	6.0	0	2	14	3	2	2.0	[5.0,27.0,6.0,0.0...	[1.34999999999999...
4	22	6.0	1	1	14	4	4	3.2666645	[4.0,22.0,6.0,1.0...	[0.34999999999999...
4	37	9.0	2	2	14	3	6	1.3611107	[4.0,37.0,9.0,2.0...	[0.34999999999999...
4	27	6.0	1	1	12	3	5	2.0	[4.0,27.0,6.0,1.0...	[0.34999999999999...
1	37	23.0	6	4	14	5	2	1.826086	[1.0,37.0,23.0,6...	[-2.65000000000000...
2	42	23.0	2	2	20	4	4	1.826086	[2.0,42.0,23.0,2...	[-1.65000000000000...
4	37	6.0	0	2	16	5	4	2.041666	[4.0,37.0,6.0,0.0...	[0.34999999999999...

```
scaled_data.select('scaled_features').show(truncate=False)
```

```
scaled_features
```

```
[[ -0.6500000000000004, 1.2500000000000036, -2.0499999999999999, 0.7500000000000004, 1.1500000000000004, 2.5500000000000007, -1.35, 0.9000000000000004]
[-0.6500000000000004, -3.7499999999999964, 1.950000000000001, 0.7500000000000004, -0.8499999999999996, -0.4499999999999993, -0.3500000000000001, -0.0999999999999964]
[0.3499999999999964, -8.749999999999996, -8.549999999999999, -2.2499999999999996, -0.8499999999999996, 1.5500000000000007, -0.3500000000000001, 0.9000000000000004]
[0.3499999999999964, 6.2500000000000036, 5.450000000000001, 1.7500000000000004, 1.1500000000000004, 1.5500000000000007, 1.65, 0.9000000000000004]
[1.349999999999996, -3.7499999999999964, -2.0499999999999999, -1.2499999999999996, -0.8499999999999996, -0.4499999999999993, -0.3500000000000001, -0.0999999999999964]
[0.3499999999999964, -3.7499999999999964, -2.0499999999999999, -2.2499999999999996, 0.1500000000000036, -0.4499999999999993, -0.3500000000000001, -0.0999999999999964]
[1.349999999999996, 6.2500000000000036, 11.950000000000001, 3.7500000000000004, 0.1500000000000036, -2.4499999999999993, 1.65, -0.0999999999999964]
[1.349999999999996, 6.2500000000000036, 11.950000000000001, 3.7500000000000004, 0.1500000000000036, -2.4499999999999993, -1.35, -1.099999999999996]
[-0.6500000000000004, -3.7499999999999964, -5.049999999999999, -2.2499999999999996, -0.8499999999999996, 1.5500000000000007, -0.3500000000000001, -1.099999999999996]
[-0.6500000000000004, -3.7499999999999964, -5.049999999999999, -2.2499999999999996, -0.8499999999999996, 1.5500000000000007, -0.3500000000000001, 0.9000000000000004]
[-1.6500000000000004, -3.7499999999999964, -5.049999999999999, -0.2499999999999996, -0.8499999999999996, 1.5500000000000007, -0.3500000000000001, 0.9000000000000004]
[1.349999999999996, -3.7499999999999964, -5.049999999999999, -0.2499999999999996, 1.1500000000000004, -0.4499999999999993, -0.3500000000000001, 0.9000000000000004]
[-0.6500000000000004, 6.2500000000000036, 5.450000000000001, 3.7500000000000004, -0.8499999999999996, -2.4499999999999993, -1.35, -1.099999999999996]
[1.349999999999996, -3.7499999999999964, -5.049999999999999, -2.2499999999999996, 0.1500000000000036, -0.4499999999999993, -0.3500000000000001, -2.099999999999996]
[0.3499999999999964, -8.749999999999996, -5.049999999999999, -1.2499999999999996, -0.8499999999999996, -0.4499999999999993, 0.6499999999999999, -0.0999999999999964]
[0.3499999999999964, 6.2500000000000036, -2.0499999999999999, -0.2499999999999996, 0.1500000000000036, -0.4499999999999993, -0.3500000000000001, 1.9000000000000004]
[0.3499999999999964, -3.7499999999999964, -5.049999999999999, -1.2499999999999996, -0.8499999999999996, -2.4499999999999993, -0.3500000000000001, 0.9000000000000004]
[-2.6500000000000004, 6.2500000000000036, 11.950000000000001, 3.7500000000000004, 2.1500000000000004, -0.4499999999999993, 1.65, -2.099999999999996]
[-1.6500000000000004, 11.250000000000001, 1.950000000000001, -0.2499999999999996, 0.1500000000000036, 5.550000000000001, 0.6499999999999999, -0.0999999999999964]
[0.3499999999999964, 6.2500000000000036, -5.049999999999999, -2.2499999999999996, 0.1500000000000036, 1.5500000000000007, 1.65, -0.0999999999999964]
```

---

## VI. BUCKETIZER

---

```
from pyspark.ml.feature import Bucketizer  
  
df.show(10, False)
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
|Col 1|Col 2|Col 3|Col 4|Col 5|Col 6|Col 7|Col 8|label  |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
|3    |32   |9.0  |3    |3    |17   |2    |5    |0.1111111|  
|3    |27   |13.0 |3    |1    |14   |3    |4    |3.2307692|  
|4    |22   |2.5  |0    |1    |16   |3    |5    |1.3999996|  
|4    |37   |16.5 |4    |3    |16   |5    |5    |0.7272727|  
|5    |27   |9.0  |1    |1    |14   |3    |4    |4.6666666|  
|4    |27   |9.0  |0    |2    |14   |3    |4    |4.6666666|  
|5    |37   |23.0 |6    |2    |12   |5    |4    |0.8521735|  
|5    |37   |23.0 |6    |2    |12   |2    |3    |1.826086  |  
|3    |22   |2.5  |0    |2    |12   |3    |3    |4.7999992|  
|3    |27   |6.0  |0    |1    |16   |3    |5    |1.3333333|  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
only showing top 10 rows
```

```
splits = [0.0,1.0,2.0,3.0,4.0,5.0,float("inf")]
```

```
splits    we are splitting into 5 bins or 5 categories
```

```
[0.0, 1.0, 2.0, 3.0, 4.0, 5.0, inf]
```


```
bucketizer = Bucketizer(splits=splits, inputCol="label",\  
                        outputCol="label_bins")
```

```
bucketizer
```

```
Bucketizer_b788654c444c
```

```
binned_df = bucketizer.transform(df)
```

```
binned_df.show()
```



Col 1	Col 2	Col 3	Col 4	Col 5	Col 6	Col 7	Col 8	label	label_bins
3	32	9.0	3	3	17	2	5	0.1111111	0.0
3	27	13.0	3	1	14	3	4	3.2307692	3.0
4	22	2.5	0	1	16	3	5	1.3999996	1.0
4	37	16.5	4	3	16	5	5	0.7272727	0.0
5	27	9.0	1	1	14	3	4	4.6666666	4.0
4	27	9.0	0	2	14	3	4	4.6666666	4.0
5	37	23.0	6	2	12	5	4	0.8521735	0.0
5	37	23.0	6	2	12	2	3	1.826086	1.0
3	22	2.5	0	2	12	3	3	4.7999992	4.0
3	27	6.0	0	1	16	3	5	1.3333333	1.0
2	27	6.0	2	1	16	3	5	3.2666645	3.0
5	27	6.0	2	3	14	3	5	2.0416666	2.0
3	37	16.5	6	1	12	2	3	0.4848484	0.0
5	27	6.0	0	2	14	3	2	2.0	2.0
4	22	6.0	1	1	14	4	4	3.2666645	3.0
4	37	9.0	2	2	14	3	6	1.3611107	1.0
4	27	6.0	1	1	12	3	5	2.0	2.0
1	37	23.0	6	4	14	5	2	1.826086	1.0
2	42	23.0	2	2	20	4	4	1.826086	1.0
4	37	6.0	0	2	16	5	4	2.0416666	2.0

starting with  
0 is label 0  
1 is label 1  
2 is label 2  
3 is label 3  
4 is label 4

```
binned_df.select(['label', 'label_bins']).show(10, False)
```

```
+-----+-----+
|label   |label_bins|
+-----+-----+
|0.111111|0.0       |
|3.230769|3.0       |
|1.399999|1.0       |
|0.727272|0.0       |
|4.666666|4.0       |
|4.666666|4.0       |
|0.852173|0.0       |
|1.826086|1.0       |
|4.799999|4.0       |
|1.333333|1.0       |
+-----+-----+
only showing top 10 rows
```

```
binned_df.groupBy('label_bins').count().show()
```

```
+-----+-----+
|label_bins|count|
+-----+-----+
|         0.0|    4|
|         1.0|    6|
|         4.0|    3|
|         3.0|    3|
|         2.0|    4|
+-----+-----+
```

---

## ABOUT DR. ALVIN ANG

---



Dr. Alvin Ang earned his Ph.D., Masters and Bachelor degrees from NTU, Singapore. He is a scientist, entrepreneur, as well as a personal/business advisor. More about him at [www.AlvinAng.sg](http://www.AlvinAng.sg).