Feature Scaling for Machine Learning: Understanding the Difference Between Normalization vs. Standardization

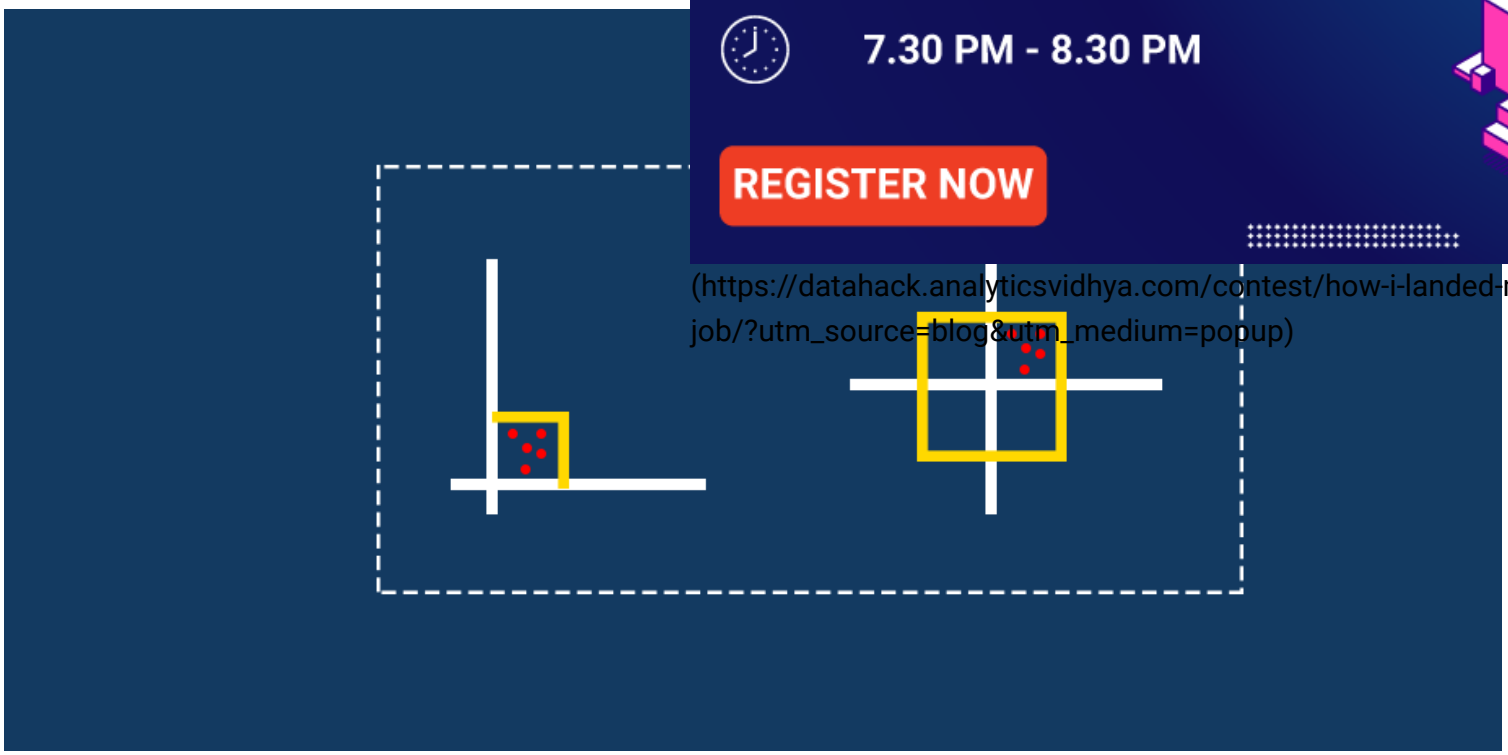
ANIRUDDHA BHANDARI (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/AUTHOR/ANIRUDDHA/), APRIL 3, 2020 LOGIN TO BOOKMARK THI...

Introduction to Feature Scaling

I was recently working with a dataset that had multiple features spanning varying degrees of magnitude, range, and units. This is a significant obstacle as a few machine learning algorithms are highly sensitive to these features.

I'm sure most of you must have faced this issue in y
feature is entirely in kilograms while the other is in g
features when they vary so vastly in terms of what t

> *This is where I turned to the conc*
> *data preprocessing stage but I've*
> *detriment of their machine learni*



(https://cdn.analyticsvidhya.com/wp-content/uploads/2020/04/Feature-image-Normalization-vs.-Standardization.png)

Here's the curious thing about feature scaling – it improves (significantly) the performance of some machine learning algorithms and does not work at all for others. What could be the reason behind this quirk? Also, what's the difference between normalization and standardization? These are two of the most commonly used feature scaling techniques in machine learning but a level of ambiguity exists in their understanding. When

should you use which technique?

I will answer these questions and more in this article on feature scaling. We will also implement feature scaling in Python to give you a practice understanding of ho

Note: I assume that you are familiar with Python an recommend going through the below courses:

- *Python for Data Science (https://courses.anal science?utm_source=blog&utm_medium=feat*
- *All free Machine Learning Courses by Analytic* /collections?category=free)
- *Applied Machine Learning (https://courses.an beginner-to-professional?utm_source=blog&u standardization)*

(https://datahack.analyticsvidhya.com/contest/how-i-landed-m job/?utm_source=blog&utm_medium=popup)

Table of Contents

Why Should we Use Feature Scaling?

The first question we need to address – why do we need to scale the variables in our dataset? Some machine learning algorithms are sensitive to feature scaling while others are virtually invariant to it. Let me explain that in more detail.

Gradient Descent Based Algorithms

Machine learning algorithms like linear regression network-from-scratch-in-python-and-r/?utm_source= normalization-standardization), logistic regression network-from-scratch-in-python-and-r/?utm_source= normalization-standardization), neural network (ht network-from-scratch-in-python-and-r/?utm_source= normalization-standardization), etc. that use gradie be scaled. Take a look at the formula for gradient de

$$\theta_j := \theta_j - \alpha \frac{1}{m}$$

(https://cdn.analyticsvidhya.com/wp-content/uploa

The presence of feature value X in the formula will a ranges of features will cause different step sizes for smoothly towards the minima and that the steps for gradient descent are updated at the same rate for all the features, we scale the data before feeding it to the model.

> *Having features on a similar scale can help the gradient descent converge more quickly towards the minima.*

Distance-Based Algorithms

Distance algorithms like KNN (https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/?utm_source=blog&utm_medium=feature-scaling-machine-learning-normalization-standardization), K-means (https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/?utm_source=blog&utm_medium=feature-scaling-machine-learning-normalization-standardization), and SVM (https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/?utm_source=blog&utm_medium=feature-scaling-machine-learning-normalization-standardization) are

most affected by the range of features. This is because behind the scenes they are using distances between data points to determine their similarity.

For example, let's say we have data containing high
their future incomes (in thousands Rupees):

	Studen
0	
1	
2	
3	
4	

(https://cdn.analyticsvidhya.com/wp-content/uploa

Since both the features have different scales, there i
higher magnitude. This will impact the performance
want our algorithm to be biassed towards one featu

> *Therefore, we scale our data before employing a distance based algorithm so that all the features contribute equally to the result.*

	Student	CGPA	Salary '000
0	1	-1.184341	1.520013
1	2	-1.184341	-1.100699
2	3	0.416120	-1.100699
3	4	1.216350	0.209657
4	5	0.736212	0.471728

(https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/knn_ex_scaled.png)

The effect of scaling is conspicuous when we compare the Euclidean distance between data points for students

A and B, and between B and C, before and after scaling as shown below:

- Distance AB before scaling =>$\sqrt{(40-60)^2+(3-3)^2}=20$
- Distance BC before scaling =>$\sqrt{(40-40)^2$
- Distance AB after scaling =>$\sqrt{(1.1+1.5)^2$
- Distance BC after scaling =>$\sqrt{(1.1-1.1)^2$

Scaling has brought both the features into the pictur
were before we applied scaling.

Tree-Based Algorithms

Tree-based algorithms (https://www.analyticsvidhya
tutorial-scratch-in-python/?utm_source=blog&utm_m
standardization), on the other hand, are fairly insens
tree is only splitting a node based on a single featur
the homogeneity of the node. This split on a feature

So, there is virtually no effect of the remaining features on the split. This is what makes them invariant to the scale of the features!

What is Normalization?

Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.

Here's the formula for normalization:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

(https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/Norm_eq.gif)

Here, Xmax and Xmin are the maximum and the minimum values of the feature respectively.

- When the value of X is the minimum value in the column, the numerator will be 0, and hence X' is 0
- On the other hand, when the value of X is the maximum value in the column, the numerator is equal to the denominator and thus the value of X' is 1
- If the value of X is between the minimum and

What is Standardization?

Standardization is another scaling technique where
standard deviation. This means that the mean of the
has a unit standard deviation.

Here's the formula for standardization:

$$X'$$

(https://cdn.analyticsvidhya.com/wp-content/uploa

μ is the mean of the feature values and σ is the star
the values are not restricted to a particular range.

Now, the big question in your mind must be when should we use normalization and when should we use standardization? Let's find out!

The Big Question – Normalize or Standardize?

Normalization vs. standardization is an eternal question among machine learning newcomers. Let me elaborate on the answer in this section.

- Normalization is good to use when you know that the distribution of your data does not follow a Gaussian distribution. This can be useful in algorithms that do not assume any distribution of the data like K-Nearest Neighbors and Neural Networks.
- Standardization, on the other hand, can be helpful in cases where the data follows a Gaussian distribution. However, this does not have to be necessarily true. Also, unlike normalization, standardization does not have a bounding range. So, even if you have outliers in your data, they will not be affected by standardization.

However, at the end of the day, the choice of using normalization or standardization will depend on your problem and the machine learning algorithm you are using. There is no hard and fast rule to tell you when to normalize or standardize your data. You can always start by fitting your model to raw, normalized and standardized data and compare the performance for best results.

It is a good practice to fit the scaler on the training d
avoid any data leakage during the model testing pro
required.

Implementing Feature Scaling in Python

Now comes the fun part – putting what we have lea
machine learning algorithms on the Big Mart datase
problem-big-mart-sales-iii/?utm_source=blog&utm_
standardization) I've taken the DataHack (https://da
utm_medium=feature-scaling-machine-learning-norr

I will skip the preprocessing steps since they are out
explained in this article (https://www.analyticsvidhy
/?utm_source=blog&utm_medium=feature-scaling-machine-learning-normalization-standardization). Those steps will enable you to reach the top 20 percentile on the hackathon leaderboard so that's worth checking out!

So, let's first split our data into training and testing sets:

```
# spliting training and testing data
from sklearn.model_selection import train_test_split

X = df
y = target

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=27)
```

view raw (https://gist.github.com/aniruddha27/02432c8623b5e202f79c3964bd559867/raw/a423c2def59d31259b0b1e358c0700112a4b42e1/NormalizationVsStandarization_1.py) NormalizationVsStandarization_1.py (https://gist.github.com/aniruddha27/02432c8623b5e202f79c3964bd559867#file-normalizationvsstandarization_1-py) hosted with ❤ by GitHub (https://github.com)

Before moving to the feature scaling part, let's glance at the details about our data using the .describe() method:

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Outlet_Size	Tier 2	Tier 3	Supermarket Type1	Supermarket Type2
count	6818.000000	6818.000000	6818.000000	6818.000000	6818.000000	6818.000000	6818.000000	6818.000000	6818.000000	6818.000000
mean	12.835420	0.355676	-2.940445	140.486413						
std	4.233450	0.478753	0.791551	62.067053						
min	4.555000	0.000000	-5.633875	31.290000						
25%	9.300000	0.000000	-3.467944	93.385700						
50%	12.857645	0.000000	-2.862535	142.179900						
75%	16.000000	1.000000	-2.331264	184.495000						
max	21.350000	1.000000	-1.113550	266.888400						

(https://cdn.analyticsvidhya.com/wp-content/uploa

We can see that there is a huge difference in the ran
Item_Visibility , Item_Weight, Item_MRP, and Outlet
scaling!

*Note: You will notice negative values in the Item_Vis
deal with the skewness in the feature.*

Normalization using sklearn

To normalize your data, you need to import the *MinMaxScalar* from the sklearn (https://courses.analyticsvidhya.com/courses/get-started-with-scikit-learn-sklearn?utm_source=blog&utm_medium=feature-scaling-machine-learning-normalization-standardization) library and apply it to our dataset. So, let's do that!

```
# data normalization with sklearn
from sklearn.preprocessing import MinMaxScaler

# fit scaler on training data
norm = MinMaxScaler().fit(X_train)

# transform training data
X_train_norm = norm.transform(X_train)

# transform testing dataabs
X_test_norm = norm.transform(X_test)
```

view raw (https://gist.github.com/aniruddha27/a41a35725ec02006bb3156e5483cb184

/raw/5a15a3fbae1a6967171185127c458674cd021f22/NormalizationVsStandarization_2.py) NormalizationVsStandarization_2.py (https://gist.github.com/aniruddha27/a41a35725ec02006bb3156e5483cb184#file-normalizationvsstandarization_2-py) hosted with ❤ by GitHub (https://github.com)

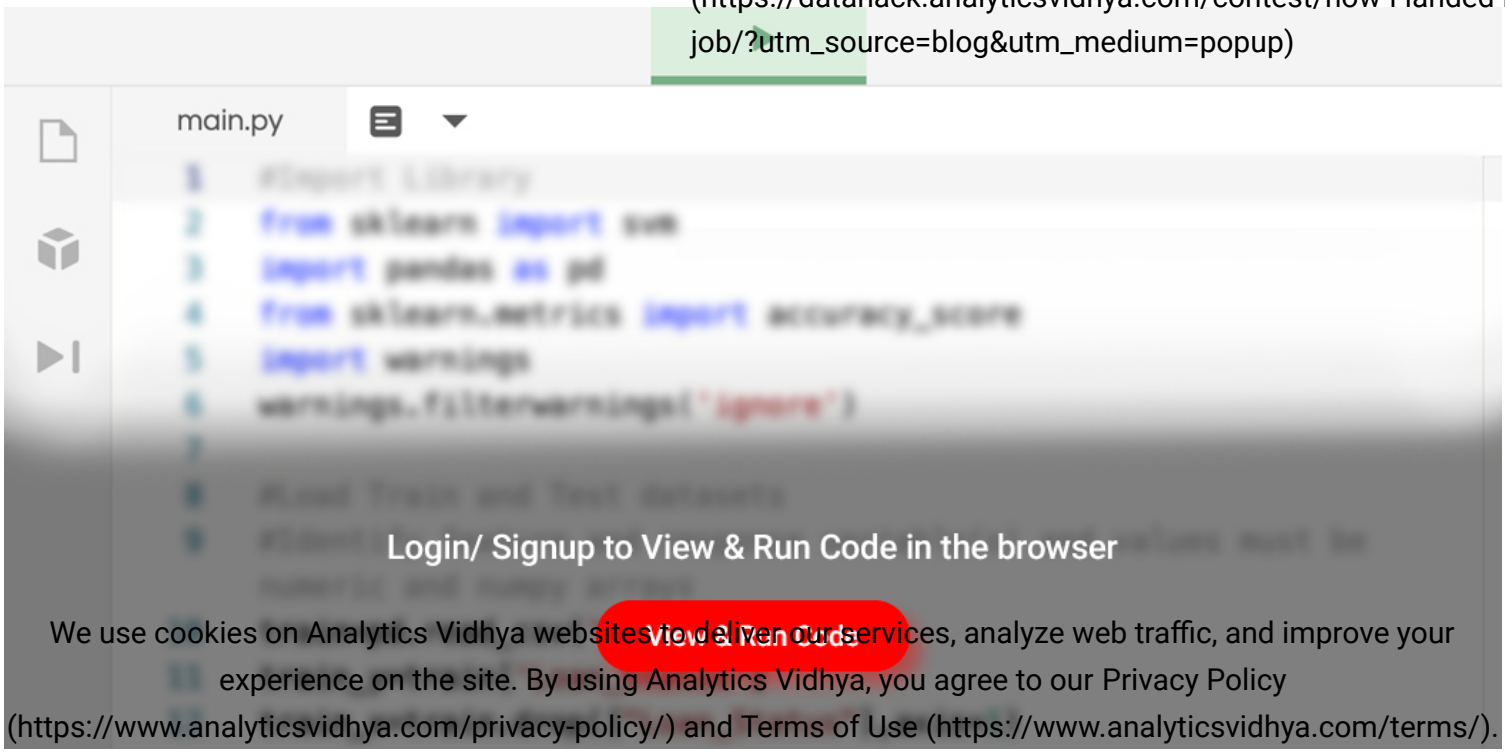
Let's see how normalization has affected our datase

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_MRP	Outle
count	6818.000000	6818.000000	6818.000000	6818.000000	
mean	0.493029	0.355676	0.595849	0.463485	
std	0.252066	0.478753	0.175109	0.263444	
min	0.000000	0.000000	0.000000	0.000000	
25%	0.282525	0.000000	0.479154	0.263566	
50%	0.494352	0.000000	0.613084	0.470673	
75%	0.681453	1.000000	0.730614	0.650280	
max	1.000000	1.000000	1.000000	1.000000	

(https://cdn.analyticsvidhya.com/wp-content/uploa

All the features now have a minimum value of 0 and

Try out the above code in the live coding window be



(https://id.analyticsvidhya.com/auth/login/?next=https://www.analyticsvidhya.com/blog/2020/04/feature-

scaling-machine-learning-normalization-standardization/?&utm_source=coding-window-blog&source=coding-window-blog)

Next, let's try to standardize our data.

Standardization using sklearn

To standardize your data, you need to import the *Sta* dataset. Here's how you can do it:

```
# data standardization with  sklearn
from sklearn.preprocessing import StandardScal

# copy of datasets
X_train_stand = X_train.copy()
X_test_stand = X_test.copy()

# numerical features
num_cols = ['Item_Weight','Item_Visibility','Item_MRP','Outlet_Establishment_Year']

# apply standardization on numerical features
for i in num_cols:

    # fit on training data column
    scale = StandardScaler().fit(X_train_stand[[i]])

    # transform the training data column
    X_train_stand[i] = scale.transform(X_train_stand[[i]])

    # transform the testing data column
    X_test_stand[i] = scale.transform(X_test_stand[[i]])
```

view raw (https://gist.github.com/aniruddha27/965ff8b01e19de1cffdb5cbe703d5495/raw/948e4d05a4e65291ff54f6986f552ea398af89de/NormalizationVsStandarization_3.py) NormalizationVsStandarization_3.py (https://gist.github.com/aniruddha27/965ff8b01e19de1cffdb5cbe703d5495#file-normalizationvsstandarization_3-py) hosted with ❤ by GitHub (https://github.com)

You would have noticed that I only applied standardization to my numerical columns and not the other One-Hot Encoded (https://www.analyticsvidhya.com/blog/2020/03/one-hot-encoding-vs-label-encoding-using-scikit-

learn/?utm_source=blog&utm_medium=feature-scaling-machine-learning-normalization-standardization) features. Standardizing the One-Hot encoded features would mean assigning a distribution to categorical features. You don't want to do that!

But why did I not do the same while normalizing the
range between 0 to 1. So, normalization would not a

Right, let's have a look at how standardization has tr

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_MRP	Ou
count	6.818000e+03	6818.000000	6.818000e+03	6.818000e+03	
mean	1.704754e-16	0.355676	2.342737e-16	1.233002e-16	
std	1.000073e+00	0.478753	1.000073e+00	1.000073e+00	
min	-1.956094e+00	0.000000	-3.402972e+00	-1.759459e+00	
25%	-8.351767e-01	0.000000	-6.664603e-01	-7.589239e-01	
50%	5.250371e-03	0.000000	9.843446e-02	2.728679e-02	
75%	7.475728e-01	1.000000	7.696596e-01	7.091011e-01	
max	2.011410e+00	1.000000	2.308161e+00	2.036689e+00	

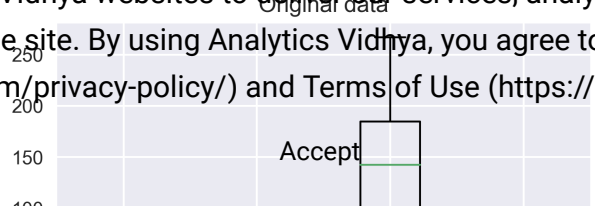
(https://cdn.analyticsvidhya.com/wp-content/uploa

The numerical features are now centered on the mean with a unit standard deviation. Awesome!

Comparing unscaled, normalized and standardized data

It is always great to visualize your data to understand the distribution present. We can see the comparison between our unscaled and scaled data using boxplots.

You can learn more about data visualization here (https://www.analyticsvidhya.com/blog/tag/data-visualization/?utm_source=blog&utm_medium=feature-scaling-machine-learning-normalization-standardization).



(https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/NormVsStand_box_plots-1.png)

You can notice how scaling the features brings everything into perspective. The features are now more comparable and will have a similar effect on the learning models.

Applying Scaling to Machine Learning Algorithms

It's now time to train some machine learning algorithms on our data to compare the effects of different scaling techniques on the performance of the algorithm. I want to see the effect of scaling on three algorithms in particular: K-Nearest Neighbours, Support Vector Regressor, and Decision Tree.

K-Nearest Neighbours

Like we saw before, KNN is a distance-based algorithm that is affected by the range of features. Let's see how it performs on our data, before and after scaling:


```
# training a KNN model
from sklearn.neighbors import KNeighborsRegres
# measuring RMSE score
from sklearn.metrics import mean_squared_erro

# knn
knn = KNeighborsRegressor(n_neighbors=7)

rmse = []

# raw, normalized and standardized training a
trainX = [X_train, X_train_norm, X_train_stan
testX = [X_test, X_test_norm, X_test_stand]

# model fitting and measuring RMSE
for i in range(len(trainX)):

    # fit
    knn.fit(trainX[i],y_train)
    # predict
    pred = knn.predict(testX[i])
    # RMSE
    rmse.append(np.sqrt(mean_squared_error(y_test,pred)))

# visualizing the result
df_knn = pd.DataFrame({'RMSE':rmse},index=['Original','Normalized','Standardized'])
df_knn
```

view raw (https://gist.github.com/aniruddha27/66119a2050fc808d2bdb7d4544ae75b6/raw/f7d5d7854dfc734b910aefc9513ab7e3140c175e/NormalizationVsStandarization_4.py) NormalizationVsStandarization_4.py (https://gist.github.com/aniruddha27/66119a2050fc808d2bdb7d4544ae75b6#file-normalizationvsstandarization_4-py) hosted with ❤ by GitHub (https://github.com)

	RMSE
Original	1319.283626
Normalized	1174.205859

Standardized	1183.448734

(https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/NormVsStand_knn.png)

You can see that scaling the features has brought d
normalized data performs a tad bit better than the s

Note: I am measuring the RMSE here because this c

Support Vector Regressor

SVR (https://www.analyticsvidhya.com/blog/2020/0
/?utm_source=blog&utm_medium=feature-scaling-m
distance-based algorithm. So let's check out whethe

```
# training an SVR model
from  sklearn.svm import SVR
# measuring RMSE score
from sklearn.metrics import mean_squared_erro

# SVR
svr = SVR(kernel='rbf',C=5)

rmse = []

# raw, normalized and standardized training and testing data
trainX = [X_train, X_train_norm, X_train_stand]
testX = [X_test, X_test_norm, X_test_stand]

# model fitting and measuring RMSE
for i in range(len(trainX)):

    # fit
    svr.fit(trainX[i],y_train)
    # predict
    pred = svr.predict(testX[i])
    # RMSE
    rmse.append(np.sqrt(mean_squared_error(y_test,pred)))

```

```
# visualizing the result
df_svr = pd.DataFrame({'RMSE':rmse},index=['Original','Normalized','Standardized'])
df_svr
```

view raw (https://gist.github.com/aniruddha27/a49f58527ef0/raw/00949b28470a33a67eca94629e52f586cacab13d/NormalizationVsStandarization_5.py) (https://gist.github.com/normalizationvsstandarization_5-py) hosted with ❤ by GitHu

Ori	
Norma	
Standard	

(https://cdn.analyticsvidhya.com/wp-content/uploa

We can see that scaling the features does bring dow
performed better than the normalized data. Why do

The sklearn documentation (https://scikit-learn.org/
mean-removal-and-variance-scaling) states that SV
centered around zero and variance is of the same order. This is because a feature with a variance greater than that of others prevents the estimator from learning from all the features. Great!

Decision Tree

We already know that a Decision tree is invariant to feature scaling. But I wanted to show a practical example of how it performs on the data:

```
# training a Decision Tree model
from sklearn.tree import DecisionTreeRegressor
# measuring RMSE score
from sklearn.metrics import mean_squared_error

# Decision tree
dt = DecisionTreeRegressor(max_depth=10,random_state=27)

rmse = []
```

```
10
11  # raw, normalized and standardized training and testing data
12  trainX = [X_train,X_train_norm,X_train_stand]
13  testX = [X_test,X_test_norm,X_test_stand]
14
15  # model fitting and measuring RMSE
16  for i in range(len(trainX)):
17
18      # fit
19      dt.fit(trainX[i],y_train)
20      # predict
21      pred = dt.predict(testX[i])
22      # RMSE
23      rmse.append(np.sqrt(mean_squared_error(y_
24
25  # visualizing the result
26  df_dt = pd.DataFrame({'RMSE':rmse},index=['Or
27  df_dt
```

view raw (https://gist.github.com/aniruddha27/6734a5390dc
/raw/049c34d17ffa4889edda3b89c7827dcf97e5b498/Normal
NormalizationVsStandarization_6.py (https://gist.github.com/
normalizationvsstandarization_6-py) hosted with ❤ by GitHu

	RMSE
Original	1245.37439
Normalized	1245.37439
Standardized	1245.37439

(https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/NormVsStand_dt.png)

You can see that the RMSE score has not moved an inch on scaling the features. So rest assured when you are using tree-based algorithms on your data!

End Notes

This tutorial covered the relevance of using feature scaling on your data and how normalization and standardization have varying effects on the working of machine learning algorithms

Keep in mind that there is no correct answer to when to use normalization over standardization and vice-versa. It all depends on your data and the algorithm you are using.

As a next step, I encourage you to try out feature sca
normalization or standardization? I recommend you
(https://datahack.analyticsvidhya.com/contest/prac
utm_medium=feature-scaling-machine-learning-norr
continuity with this article. And don't forget to share

You can also read this article on our Mobile APP

/apps/details?id=com.analyticsvidhya.android&utm

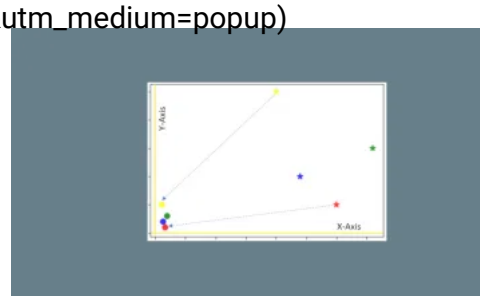
pcampaignid=MKT-Other-global-all-co-prtnr-py-PartE

(https://apps.apple.com/us/app/analytics-vidhya/id

Related Articles

(https://www.analyticsvidhya.com/blog/2020/12/feature-engineering-feature-improvements-scaling/)
Feature Engineering (Feature Improvements - Scaling)
(https://www.analyticsvidhya.com/blog/2020/12/feature-engineering-feature-improvements-scaling/)

(https://www.analyticsvidhya.com/blog/2020/10/7-feature-engineering-techniques-machine-learning/)
7 Feature Engineering Techniques in Machine Learning You Should Know
(https://www.analyticsvidhya.com/blog/2020/10/7-feature-engineering-techniques-machine-learning/)



(https://www.analyticsvidhya.com/blog/2020/07/types-of-feature-transformation-and-scaling/)
Feature Transformation and Scaling Techniques to Boost Your Model Performance
(https://www.analyticsvidhya.com/blog/2020/07/types-of-feature-transformation-and-scaling/)

TAGS : FEATURE SCALING (HTTPS://WWW.ANALYTICSVIDHY
LEARNING (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/T
PYTHON (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG
(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/LIVE-CO
(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/NORMA
(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/NORMA
(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/STANDA

N

Supervised Learning vs. Unsupervis

(https://www.analyticsvidhya.com/blog/2

PREVIOUS ARTICLE

6 Open Source Data Science Projects to Make you Industry Ready!

(https://www.analyticsvidhya.com/blog/2020/04/6-open-source-data-science-projects-industry-ready/)

(https://www.analyticsvidhya.com/blog/author/aniruddha/)

Aniruddha Bhandari (Https://www.analyticsvidhya.com/blog/author/aniruddha/)

I am on a journey to becoming a data scientist. I love to unravel trends in data, visualize it and predict the future with ML algorithms! But the most satisfying part of this journey is sharing my learnings, from the challenges that I face, with the community to make the world a better place!

This article is quite old and you might not get a pr
this comment on Analytics Vidhya's Discussion p
queries resolved

(https://datahack.analyticsvidhya.com/contest/how-i-landed-m
job/?utm_source=blog&utm_medium=popup)

16 COMMENTS

ALI

April 12, 2020 at 11:18 pm (https://www.analyticsvi
standardization/#comment-161122)

Excelent article! Thank you very much for sharing. I
to fit the scaler on the training data and then use it t
you posted. Am I wrong? How would one “fit the sca
testing data”? Thanks a lot again

ANIRUDDHA BHANDARI **Reply**

April 13, 2020 at 11:44 am (https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/#comment-161126)

Hi Ali
You fit the scaler on the training data so that it can calculate the necessary parameters, like mean and standard deviation for standardization, and store it for later use using the fit() method. Later you use the transform() function to apply the same transformation on both, train and test dataset.
I have used this approach for both, normalization and standardization, in the article in the gists “NormalizationVsStandarization_2.py” and “NormalizationVsStandarization_3.py” respectively.
I hope this cleared your doubt.
Thanks

SAHIL KAMBOJ **Reply**

April 28, 2020 at 4:25 pm (https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-

standardization/#comment-161208)

Good article! Thank you very much for sharing. I have one question.
What the difference between sklearn.preprocessing
sklearn.preprocessing.Normalizer?
When to use MinMaxScaler and when to Normalize?

ANIRUDDHA BHANDARI

April 29, 2020 at 2:15 pm (https://www.analyticsvid
standardization/#comment-161217)

Hi
I hope MinMaxScaler is already clear from the articl
Normalizer (https://scikit-learn.org/stable/modules/
/sklearn.preprocessing.Normalizer.html#sklearn.pre
The only difference is the way it computes the norm
row values i.e. each element of a row is normalized
elements in that row.
As mentioned in the documentation, it is useful in te
(https://www.analyticsvidhya.com/blog/2020/02/qu
cosine similarity between the different sentences/documents in the dataset. Other than that, as I mentioned in the article, there is no sure way to know which scaling technique should be used when. The best way is to create multiple scaled copies of the data and then try them out and see which one gives the best result.
Hope this helps.

SUBHASH KUMAR NADAR Reply

May 23, 2020 at 11:25 am (https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/#comment-161548)

Excellent article! Easy to understand and good coverage
One question: I see that there is a scale() funtion as well from sklearn and short description suggest it to be similar to StandardScaler i.e. scaling to unit variance
I could not find more than this explanation. Please can you suggest which on to use which scenario?
Thanks in advance!

ANIRUDDHA BHANDARI Reply

May 24, 2020 at 12:41 pm (https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-

standardization/#comment-161579)

Hi Subhash
I notice two differences between the two functions.
your data along any axis. This means that you could
wise, which is what happens in StandardScaler(). Se
transform methods, so you cannot apply the same s
I would suggest using the StandardScaler() function
I hope this helps!

GOLLA KEDARKUMAR
May 24, 2020 at 9:56 am (https://www.analyticsvidh
standardization/#comment-161575)

Hi ANIRUDDHA,
If we use the same scaler for train and testing, does
need to use the mean of the data. If we take the mea
the test data, right?

ANIRUDDHA BHANDARI Reply
May 24, 2020 at 12:11 pm (https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/#comment-161578)

Scaling your test data according to the train data makes sure that the test data is on the same scale as the training data on which our model was trained on. This way our model will be able to apply the learnings from the training dataset on the testing dataset, which is exactly what we want! If instead, we scale the test data differently, then our model might not be able to discern that difference, thereby giving us incorrect outputs. That way we will never know how well our model is performing.
I hope this helps!

INAS Reply
May 27, 2020 at 6:42 am (https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/#comment-161630)

Excellent article, thank you for sharing.

SOUMADIP ROY Reply

July 4, 2020 at 12:59 am (https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/#comment-162136)

This is an excellent write up. Thanks for this.

HARSHVARDHAN BHATT

July 5, 2020 at 3:03 am (https://www.analyticsvidhy
standardization/#comment-162151)

That graphs really helps in putting things in perspec

ARNOB

July 5, 2020 at 4:13 pm (https://www.analyticsvidhy
standardization/#comment-162155)

Hey bro! Great article. It covered a lots of topics that
How can I check my data after normalization. You h
sklearn' section. But when I use it I get an error – " n

Can you tell me how to check my data after normalization?

Thank you for your time.

ANIRUDDHA BHANDARI Reply

August 22, 2020 at 8:06 pm (https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/#comment-162806)

Hi Arnob, glad you liked the article.
The command you are looking for is df.describe() not pd.describe(). Try using that, it should work.

DEEPS **Reply**

July 15, 2020 at 7:59 pm (https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/#comment-162296)

Excellent article !

KUNAL

July 31, 2020 at 1:19 pm (https://www.analyticsvidh
standardization/#comment-162543)

Thanks for Great Article..!!!

ZINEB

August 17, 2020 at 3:29 pm (https://www.analytics
standardization/#comment-162737)

Thanks Bhandari.
Easy to understand and very helpful.

(https://datahack.analyticsvidhya.com/contest/how-i-landed-my-job/?utm_source=blog&utm_medium=popup)

POPULAR POSTS

Understanding Delimiters in Pandas read_csv() F
/04/delimiters-in-pandas-read_csv-function/)

40 Questions to test a Data Scientist on Clusterin
(https://www.analyticsvidhya.com/blog/2017/02/test-data-scientist-clustering/)

How to Download Kaggle Datasets using Jupyter Notebook (https://www.analyticsvidhya.com/blog/2021/04/how-to-download-kaggle-datasets-using-jupyter-notebook/)

Python List Programs For Absolute Beginners (https://www.analyticsvidhya.com/blog/2021/04/python-list-programs-for-absolute-beginners/)

Commonly used Machine Learning Algorithms (with Python and R Codes) (https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/)

30 Questions to test a data scientist on K-Nearest Neighbors (kNN) Algorithm (https://www.analyticsvidhya.com/blog/2017/09/30-questions-test-k-nearest-neighbors-algorithm/)

45 Questions to test a data scientist on basics of Deep Learning (along with solution) (https://www.analyticsvidhya.com/blog/2017/01/must-know-questions-deep-learning/)

Introductory guide on Linear Programming for (aspiring) data scientists (https://www.analyticsvidhya.com/blog/2017/02/lintroductory-guide-on-linear-programming-explained-in-simple-english/)

CAREER RESOURCES

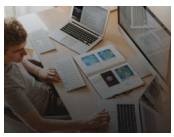
16 Key Que
Science (h
science-ca
utm_mediu

NOVEMBER 23, 2020

(https://datahack.analyticsvidhya.com/contest/how-i-landed-my job/?utm_source=blog&utm_medium=popup)

Here's What You Need to Know to Become a Data Scientist! (https://www.analyticsvidhya.com/blog/2021/01/heres-what-you-need-to-know-to-become-a-data-scientist/?&utm_source=Blog&utm_medium=CareerResourceWidget)

JANUARY 22, 2021



These 7 Signs Show you have Data Scientist Potential! (https://www.analyticsvidhya.com/blog/2020/12/these-7-signs-show-you-have-data-scientist-potential/?&utm_source=Blog&utm_medium=CareerResourceWidget)

DECEMBER 3, 2020

How To Have a Career in Data Science (Business Analytics)? (https://www.analyticsvidhya.com/blog/2020/11/how-to-have-a-career-in-data-science-business-analytics/?&utm_source=Blog&utm_medium=CareerResourceWidget)

NOVEMBER 26, 2020

Should I become a data scientist (or a business analyst)? (https://www.analyticsvidhya.com/blog/2020/11/become-data-scientist-b
utm_mediu

NOVEMBER 2

(https://datahack.analyticsvidhya.com/contest/how-i-landed-m
job/?utm_source=blog&utm_medium=popup)

RECENT POSTS

**Complete Guide on Encoding Numerical Feature
/blog/2021/05/complete-guide-on-encode-numeri**

MAY 12, 2021

**How AI Supports Logistics Industry and Transpo
/blog/2021/05/how-ai-supports-logistics-industry**

MAY 11, 2021

**Making Programming with Date and Time, less p
/05/making-programming-with-date-and-time-less-painless/)**

MAY 11, 2021

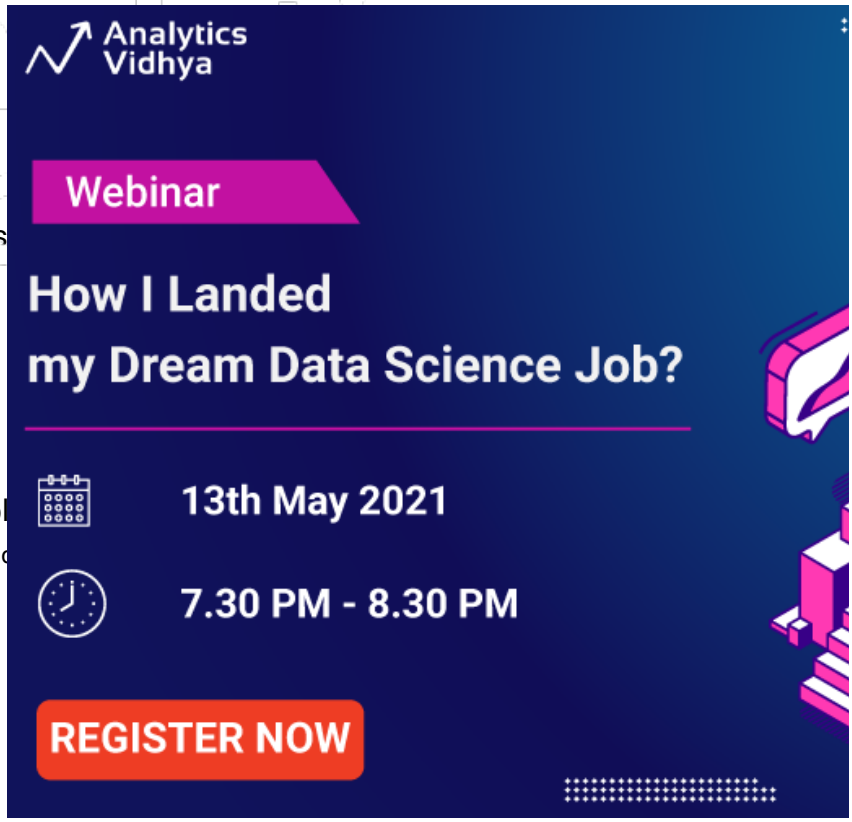
Is there any need of Deep Learning? (https://www.analyticsvidhya.com/blog/2021/05/is-there-any-need-of-deep-learning/)

MAY 11, 2021

(https://bootcamp.analyticsvidhya.com/?utm_source=Blog&

utm_medium=stickybanner1#av-roadmap)

(https://www.analyticsvidhya.com/)



(https://datahack.analyticsvidhya.com/contest/how-i-landed-my-job/?utm_source=blog&utm_medium=popup)

Download App (https://play.google.com/store/apps/details?id=com.analyticsvidhya.android

blogathon-7/?utm_source=sticky2&utm_medium=bl

Analytics Vidhya

About Us (https://www.analyticsvidhya.com/about-me/)

Our Team (https://www.analyticsvidhya.com/about-me/team/)

Careers (https://www.analyticsvidhya.com/about-me/career-analytics-vidhya/)

Contact us (https://www.analyticsvidhya.com/contact/)

Discussions (https://discuss.analyticsvidhya.com/)

Apply Jobs (https://www.analyticsvidhya.com/jobs/)

Companies

Post Jobs (https://www.analyticsvidhya.com/corporate/)

Trainings (https://courses.analyticsvidhya.com/)

Hiring Hackathons (https://datahack.analyticsvidhya.com/)

Advertising (https://www.analyticsvidhya.com/contact/)

Visit us

(https://www.linkedin.com/company/analytics-vidhya/) (https://www.youtube.com/channel/UCkw4JCwteGrDHIsyIIKo4tQ) (https://www.facebook.com/analyticsvidhya) (https://twitter.com/Analytics_Vidhya?lang=en) (https://www.instagram.com/analyticsvidhya/?hl=en)