Data Transformation

Data in different scales.

Values in a dataset might have a variety of different magnitudes, ranges, or scales. Algorithms that use distance as a parameter may not weigh all these in the same way. There are various data transformation techniques that are used to transform the features of our data so that they use the same scale, magnitude, or range. This ensures that each feature has an appropriate effect on a model's predictions. Some features in our data might have high-magnitude values (for example, annual salary), while others might have relatively low values (for example, the number of years worked at a company). Just because some data has smaller values does not mean it is less significant.

Reference: Data Science with Python By Rohan Chopra, Aaron England, Mohamed Noordeen Alaudeen July 2019

https://subscription.packtpub.com/book/data/9781838552862/1/ch01lvl1sec08/data-in-different-scales

Implementing Scaling Using the Standard Scaler Method

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

In [ ]:
```
df = pd.read_csv(r'C:\Users\Raul\OneDrive\Escritorio\CS\TC2004B.101\data\Wholesale
```

In [ ]:
```
df.head()
```

Out[ ]:

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	2	3	12669	9656	7561	214	2674	1338
1	2	3	7057	9810	9568	1762	3293	1776
2	2	3	6353	8808	7684	2405	3516	7844
3	1	3	13265	1196	4221	6404	507	1788
4	2	3	22615	5410	7198	3915	1777	5185

In [ ]:
```
dtypes = df.dtypes
dtypes
```

Out[ ]:
```
Channel             int64
Region              int64
Fresh               int64
Milk                int64
Grocery             int64
Frozen              int64
Detergents_Paper    int64
Delicassen          int64
dtype: object
```

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 440 entries, 0 to 439
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Channel           440 non-null    int64
 1   Region            440 non-null    int64
 2   Fresh             440 non-null    int64
 3   Milk              440 non-null    int64
 4   Grocery           440 non-null    int64
 5   Frozen            440 non-null    int64
 6   Detergents_Paper  440 non-null    int64
 7   Delicassen        440 non-null    int64
dtypes: int64(8)
memory usage: 27.6 KB
```

Perform standard scaling and print the first five rows of the new dataset. To do so, use the StandardScaler() class from sklearn.preprocessing and implement the fit_transorm() method. Using the StandardScaler method, we will scale the data into a uniform unit over all the columns. The values of all the features will be converted into a uniform range of the same scale. Because of this, it becomes easier for the model to make predictions.

In [ ]:
```
from sklearn import preprocessing
std_scale = preprocessing.StandardScaler().fit_transform(df)
scaled_frame = pd.DataFrame(std_scale, columns=df.columns)

scaled_frame.head()
```

Out[ ]:

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delica
0	1.448652	0.590668	0.052933	0.523568	-0.041115	-0.589367	-0.043569	-0.06
1	1.448652	0.590668	-0.391302	0.544458	0.170318	-0.270136	0.086407	0.08
2	1.448652	0.590668	-0.447029	0.408538	-0.028157	-0.137536	0.133232	2.24
3	-0.690297	0.590668	0.100111	-0.624020	-0.392977	0.687144	-0.498588	0.09
4	1.448652	0.590668	0.840239	-0.052396	-0.079356	0.173859	-0.231918	1.29

Implementing Scaling Using the MinMax Scaler Method

Perform MinMax scaling and print the initial five values of the new dataset. To do so, use the MinMaxScaler() class from sklearn.preprocessing and implement the fit_transorm() method. Add the following code to implement this: Using the MinMaxScaler method, we will scale the data into a uniform unit over all the columns

In [ ]:
```
from sklearn import preprocessing
minmax_scale = preprocessing.MinMaxScaler().fit_transform(df)
# Esto es equivalente a
# minmax_scale = preprocessing.MinMaxScaler((0,1)).fit_transform(df)
scaled_frame = pd.DataFrame(minmax_scale,columns=df.columns)
scaled_frame.head()
```

Out[ ]:

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	1.0	1.0	0.112940	0.130727	0.081464	0.003106	0.065427	0.027847
1	1.0	1.0	0.062899	0.132824	0.103097	0.028548	0.080590	0.036984
2	1.0	1.0	0.056622	0.119181	0.082790	0.039116	0.086052	0.163559
3	0.0	1.0	0.118254	0.015536	0.045464	0.104842	0.012346	0.037234
4	1.0	1.0	0.201626	0.072914	0.077552	0.063934	0.043455	0.108093

In [ ]:
```
# (c) En el escalado de características min-max, aplíquelo para un límite entre [0,
minmax_scale = preprocessing.MinMaxScaler((-1,1)).fit_transform(df)
scaled_frame = pd.DataFrame(minmax_scale, columns=df.columns)
scaled_frame.describe()
```

Out[ ]:

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_
count	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000	440.0
mean	-0.354545	0.543182	-0.786045	-0.843654	-0.828658	-0.899844	-0.8
std	0.936103	0.774272	0.225547	0.200982	0.204860	0.159578	0.2
min	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.0
25%	-1.000000	0.000000	-0.944275	-0.959751	-0.953652	-0.976423	-0.9
50%	-1.000000	1.000000	-0.848397	-0.902727	-0.897550	-0.950661	-0.9
75%	1.000000	1.000000	-0.698064	-0.805693	-0.770358	-0.883990	-0.8
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0

In [ ]:
```
#  b) Agrega el código necesario para normalizar los datos utilizando escalamiento

def decimal(num):
    j = len(str(abs(num)))
    return num/10**j

temp = pd.DataFrame()
for column in df:
    biggest = max(abs(df[column]))
    scale = len(str(abs(biggest)))
    temp[column] = df[column]/(10**scale)
```

In [ ]:
```
display(max(df.Fresh))
temp.describe()
```

```
112151
```

Out[ ]:

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_
count	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000	440.0
mean	0.132273	0.254318	0.012000	0.057963	0.079513	0.030719	0.0
std	0.046805	0.077427	0.012647	0.073804	0.095032	0.048547	0.0
min	0.100000	0.100000	0.000003	0.000550	0.000030	0.000250	0.0
25%	0.100000	0.200000	0.003128	0.015330	0.021530	0.007423	0.0
50%	0.100000	0.300000	0.008504	0.036270	0.047555	0.015260	0.0
75%	0.200000	0.300000	0.016934	0.071903	0.106557	0.035543	0.0
max	0.200000	0.300000	0.112151	0.734980	0.927800	0.608690	0.4