SERVERLESS IOT DATA PROCESSING

Phase 3 submission:

INTRODUCTION TO LOADING DATASET :

IOT Data analysis

Anomaly Detection, Analysis based on data collected from 54 sensors

Future work: Using LSTM or GRU, we can predict future temperature,humidity,light and voltage readings based on the given time series data. It is the case of Multidimensional and multivariate time series model.

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import datetime
%matplotlib inline
import scipy.integrate as integrate
from scipy.optimize import curve_fit
pd.options.display.max_rows = 12
```

```
data = pd.read_csv("../input/data.txt", sep = ' ',header = None, names = ['date',
'time','epoch','moteid','temp','humidity','light','voltage'])
data.head()
```

Out[2]:

	date	time	epoch	moteid	temp	humidity	light	voltage

0	2004-03-31	03:38:15.757551	2	1.0	122.1530	-3.91901	11.04	2.03397
1	2004-02-28	00:59:16.02785	3	1.0	19.9884	37.09330	45.08	2.69964
2	2004-02-28	01:03:16.33393	11	1.0	19.3024	38.46290	45.08	2.68742
3	2004-02-28	01:06:16.013453	17	1.0	19.1652	38.80390	45.08	2.68742
4	2004-02-28	01:06:46.778088	18	1.0	19.1750	38.83790	45.08	2.69964

```
data.shape
```

Out[3]:

```
(2313682, 8)
```

In [4]:

```
data.describe()
```

```
data.describe()
```

Out[4]:

	epoch	moteid	temp	humidity	light	voltage
count	2.313682e+06	2.313156e+06	2.312781e+06	2.312780e+06	2.219804e+06	2.313156e+06
mean	3.303993e+04	2.854412e+01	3.920700e+01	3.390814e+01	4.072110e+02	2.492552e+00
std	1.836852e+04	5.062408e+01	3.741923e+01	1.732152e+01	5.394276e+02	1.795743e-01
min	0.000000e+00	1.000000e+00	-3.840000e+01	-8.983130e+03	0.000000e+00	9.100830e-03
25%	1.757200e+04	1.700000e+01	2.040980e+01	3.187760e+01	3.956000e+01	2.385220e+00
50%	3.332700e+04	2.900000e+01	2.243840e+01	3.928030e+01	1.582400e+02	2.527320e+00
75%	4.778900e+04	4.100000e+01	2.702480e+01	4.358550e+01	5.372800e+02	2.627960e+00
max	6.553500e+04	6.540700e+04	3.855680e+02	1.375120e+02	1.847360e+03	1.856000e+01

```
data.isnull().sum()
```

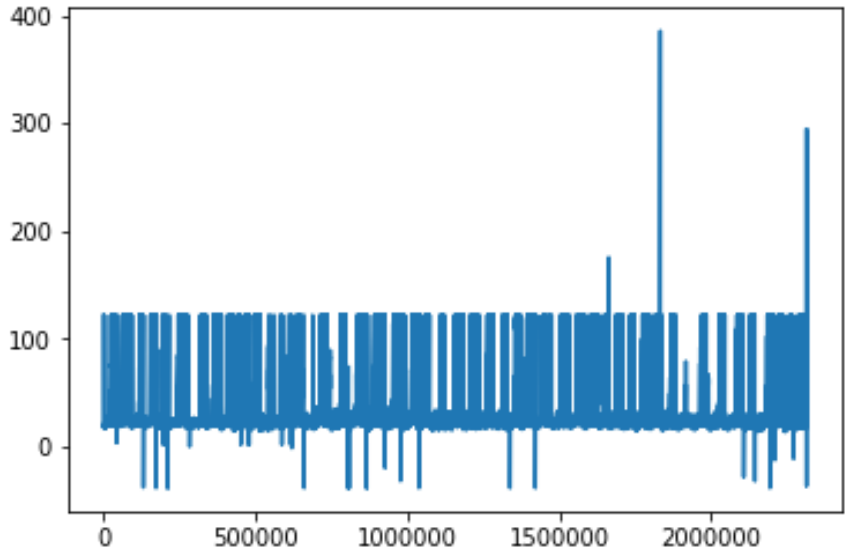
Out[5]:

```
date            0
time            0
epoch           0
moteid        526
temp          901
humidity      902
light       93878
voltage       526
dtype: int64
```

In [6]:

```
data['temp'].plot()
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f83485fb1d0>




```
data.fillna(0, inplace=True)
data['epoch'].replace(regex=True, inplace=True, to_replace=r'[^0-9.\-]', value=r'')
data['epoch'] = data['epoch'].astype(int)
```

In [8]:

```
data.isnull().sum()
```

```
date        0
time        0
epoch       0
moteid      0
temp        0
humidity    0
light       0
voltage     0
dtype: int64
```

In [9]:

```
data.groupby('moteid').mean()
```

	temp	humidity	light	voltage	
epoch					
moteid					
0.0	22031.159696	0.000000	0.000000	0.000000	0.000000
1.0	31994.793830	35.882437	34.319280	156.575828	2.519643
2.0	36491.176404	40.201817	34.298739	212.159717	2.458436
3.0	33013.771411	33.715289	34.787174	146.049674	2.517961
4.0	32688.045167	45.464410	29.991159	155.491512	2.468170
5.0	5836.428571	0.000000	0.000000	0.000000	2.516088
...	...	...	...	...	...
56.0	54586.336657	20.987656	39.261422	104.099945	2.887558
57.0	1499.000000	0.000000	17.887833	0.000000	2.482020
58.0	53830.011773	21.014753	38.541575	364.111968	2.899639
6485.0	35112.000000	262.656000	0.000000	0.000000	0.393324
33117.0	30493.000000	-36.204800	0.000000	0.000000	0.144859
65407.0	18182.000000	294.251000	0.000000	0.000000	0.013056

```
data['Timestamp'] = data[['date', 'time']].apply(lambda x: ' '.join(x.astype(str)), axis=1)
new_data = data
```

In [11]:

```
data.drop(['date','time'],axis=1,inplace =True)
data.set_index(pd.to_datetime(data.Timestamp), inplace=True)
```

In [12]:

```
data[['moteid','temp','humidity','light','voltage']] =
data[['moteid','temp','humidity','light','voltage']].apply(pd.to_numeric)
```

```
data['moteid'].value_counts()
```

Out[13]:

```
31.0       65694
29.0       64391
23.0       62440
26.0       61521
22.0       60165
21.0       58525
           ...
0.0          526
5.0           35
57.0           3
65407.0        1
6485.0         1
33117.0        1
Name: moteid, Length: 62, dtype: int64
```

moteid_grp = data.groupby(['moteid'])

In [15]:

corr_id = moteid_grp.corr(method='pearson')
corr_id.fillna(0, inplace=True)
corr_id

epoch	humidity	light	temp	voltage		
moteid						
0.0	epoch	1.0	0.000000	0.000000	0.000000	0.000000
	humidity	0.0	0.000000	0.000000	0.000000	0.000000
	light	0.0	0.000000	0.000000	0.000000	0.000000
	temp	0.0	0.000000	0.000000	0.000000	0.000000
	voltage	0.0	0.000000	0.000000	0.000000	0.000000
1.0	epoch	1.0	-0.195395	0.014605	0.315767	-0.726957
...	...	...	...	...	...	...
33117.0	voltage	0.0	0.000000	0.000000	0.000000	0.000000

65407.0	epoch	0.0	0.000000	0.000000	0.000000	0.000000
	humidity	0.0	0.000000	0.000000	0.000000	0.000000
	light	0.0	0.000000	0.000000	0.000000	0.000000
	temp	0.0	0.000000	0.000000	0.000000	0.000000
	voltage	0.0	0.000000	0.000000	0.000000	0.000000

310 rows × 5 columns

epoch	moteid	temp	humidity	light	voltage	Timestamp	
Timestamp							
2004-03-31 03:38:15.757551	2	1.0	122.1530	-3.91901	11.04	2.03397	2004-03-31 03:38:15.757551
2004-02-28 00:59:16.027850	3	1.0	19.9884	37.09330	45.08	2.69964	2004-02-28 00:59:16.02785
2004-02-28 01:03:16.333930	11	1.0	19.3024	38.46290	45.08	2.68742	2004-02-28 01:03:16.33393
2004-02-28 01:06:16.013453	17	1.0	19.1652	38.80390	45.08	2.68742	2004-02-28 01:06:16.013453
2004-02-28 01:06:46.778088	18	1.0	19.1750	38.83790	45.08	2.69964	2004-02-28 01:06:46.778088

Temperature_data

de22	Node23	Node24	Node25	Node26	Node27	Node28	Node29					
Timestamp	Timestamp	Timestamp	Timestamp									
2004	3	1	0	18.892649	17.888361	18.682962	17.909755	17.859044	17.673663	18.159542	16.981087	17.684646
			1	17.828778	17.513253	18.301935	17.535843	17.494400	17.259217	17.705163	16.576858	17.429180
			2	17.478076	17.269043	18.038685	17.276184	17.222923	16.988568	17.443851	16.263400	17.193869
			3	17.265336	17.036493	17.833192	17.068284	17.011023	16.778655	17.242236	16.098262	17.046103
			4	16.976067	16.728080	17.526390	16.730619	16.708245	16.424560	16.829928	15.745938	16.794936
			5	16.678958	16.295632	17.130163	16.236195	16.223800	15.896133	16.312522	15.345038	16.430650
		...	...	...	...	...	...	...	...	...	...	...

		21	18	21.580820	20.955944	24.125349	32.503942	21.184272	21.099364	22.016777	20.406817	21.141881
			19	21.213658	20.334054	24.385489	35.667856	20.598625	20.551900	21.721577	19.656331	20.547413
			20	20.893717	19.951263	24.994137	40.132790	20.216815	20.256095	21.630513	19.215485	20.144796
			21	20.408241	19.569963	25.837693	45.156295	19.830846	19.937289	21.666446	18.845821	19.723914
			22	20.171971	19.167160	27.153425	51.666562	19.390084	19.724252	21.884614	18.476750	19.241492
			23	19.997640	18.620570	28.617012	59.078113	18.886997	19.474558	22.185723	17.960217	18.716850

463 rows × 9 columns

In [19]:

Humidity_data

Out[19]:

				Node14	Node22	Node23	Node24	Node25	Node26	Node27	Node28	Node29
Timestamp	Timestamp	Timestamp	Timestamp									
2004	3	1	0	42.549285	44.530996	43.168231	44.878085	44.633242	47.021085	45.492716	48.549510	46.209795
			1	45.355191	45.693293	44.364838	46.024384	45.775104	48.405524	46.959683	49.996614	46.909245
			2	46.163976	46.084623	44.710712	46.487651	46.130820	48.831494	47.188011	50.559377	47.271673
			3	46.613439	46.469025	45.070823	46.854633	46.512194	49.220666	47.632574	50.843643	47.565288
			4	47.179816	47.056620	45.602291	47.544280	47.070312	49.981491	48.445950	51.645920	48.041444
			5	47.897721	48.318876	46.838235	49.095143	48.524784	51.741095	50.494748	52.944230	49.184805
		...	...	...	...	...	...	...	...	...	...	...
		21	18	48.074626	47.581688	46.317189	48.043332	45.830167	48.557737	46.804405	48.878659	46.799120
			19	48.199534	48.236579	46.810787	48.842767	46.393657	49.506822	47.598095	50.102531	47.288588
			20	48.285621	48.535250	47.076702	49.253592	46.754700	49.895563	48.139502	50.566962	47.463996
			21	49.033441	48.811621	47.416511	49.718356	47.142598	50.570302	48.718429	50.572297	47.832622
			22	49.483676	49.228436	47.773140	50.317610	47.654925	51.163117	49.398291	50.813785	48.433595
			23	49.850957	50.196400	48.584533	51.448840	48.664630	52.343430	50.653467	51.896545	49.429506

Covariance models in time for the temperature

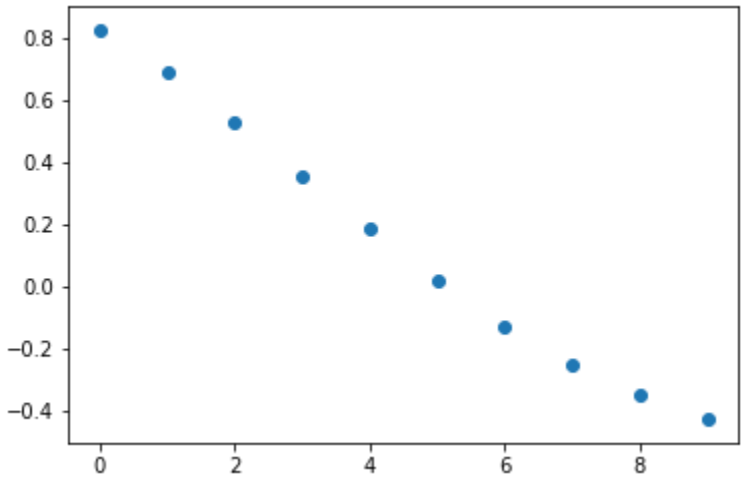
Temperature time lags from 0 to 9

In [20]:

```
node_distance = pd.read_csv('../input/intel_nodes_distances.csv')

lists_hy = []
lists_hx = []

for z in range(10):
    # Calculation of the covariance for lag i hour for temp
    humy = [None] *45
    humx = [None] *45
    k=0
    i=0
    for first_column in Temperature_data:
        df1 = Temperature_data[first_column][:-(1+z)]
        std_test1=np.std(df1)
        j = 0
        for second_column in Temperature_data:
            if j <= i:
                df2 = Temperature_data[second_column][(1+z):]
                std_test2=np.std(df2)
                humy[k] = (np.cov(df1,df2)/(std_test1*std_test2)).item((0, 1))
                humx[k] = node_distance.iloc[i,j]
                j = j + 1
                k = k + 1
        i = i + 1
    lists_hy.append(humy)
    lists_hx.append(humx)
```

Out[21]:

<matplotlib.collections.PathCollection at 0x7f8327bc9550>

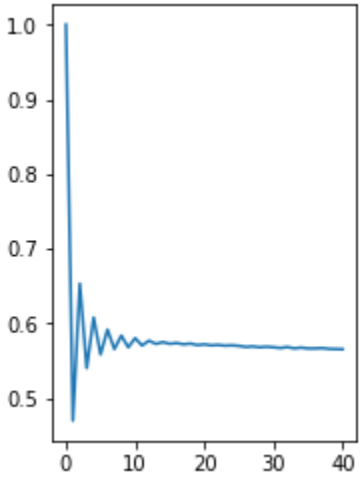


In [22]:

```
from statsmodels.tsa.stattools import acf,pacf
lag_acf = acf(data['temp'])
#Plot pACF: a
plt.subplot(121)
plt.plot(lag_acf)
```

Out[22]:

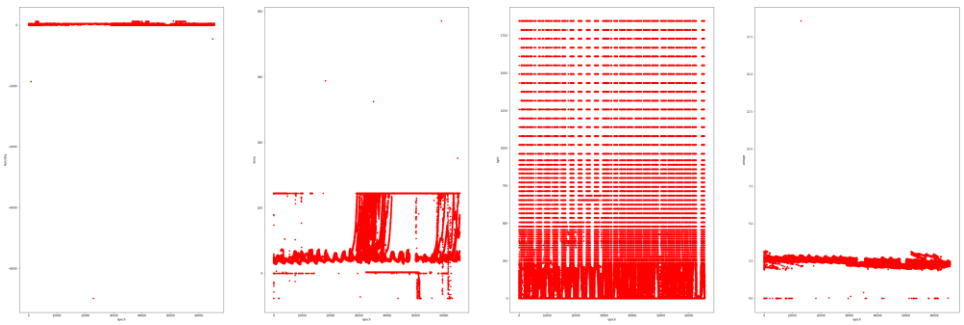
```
[<matplotlib.lines.Line2D at 0x7f8326a94470>]
```



Variation in humidity, temp, light, voltage with epoch

In [23]:

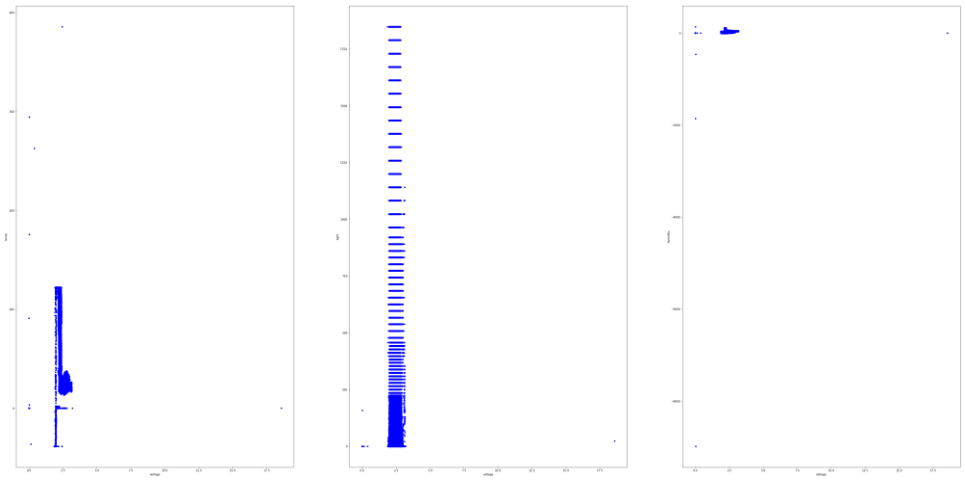
```
fig, axes = plt.subplots(nrows=1, ncols=4, figsize=(60,20))

for xcol, ax in zip(['humidity', 'temp', 'light','voltage'], axes):
    data.plot(kind='scatter', x='epoch', y=xcol, ax=ax, alpha=1, color='r')
```



Variation with Voltage

In [24]:

```
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(60,30))
for xcol, ax in zip(['temp', 'light','humidity'], axes):
    data.plot(kind='scatter', x='voltage', y=xcol, ax=ax, color='b')
```



Variation with light

In [25]:

```
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(60,30))
for xcol, ax in zip(['temp', 'humidity','voltage'], axes):
    data.plot(kind='scatter', x='light', y=xcol, ax=ax, alpha=1, color='g')
```

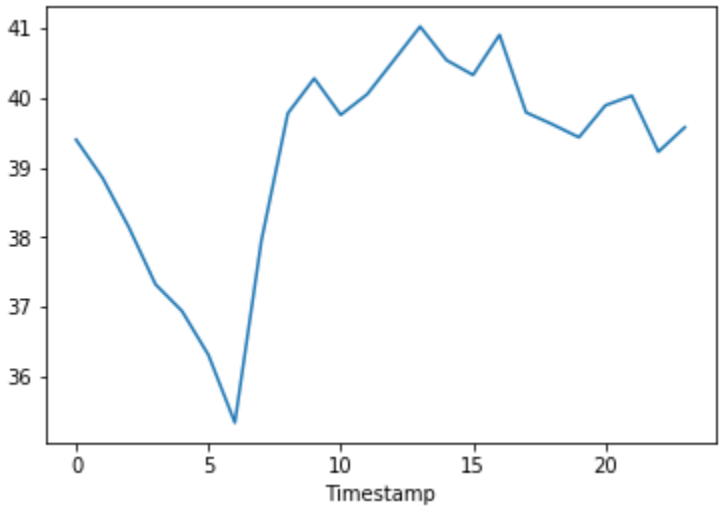
Variation with Humidity

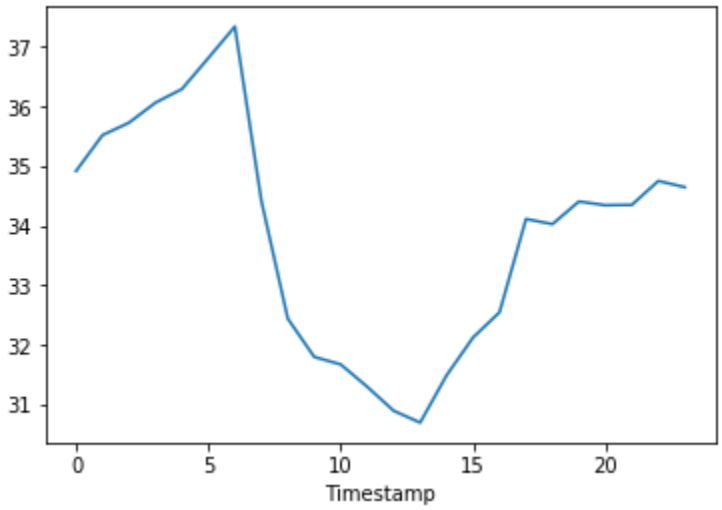
In [26]:

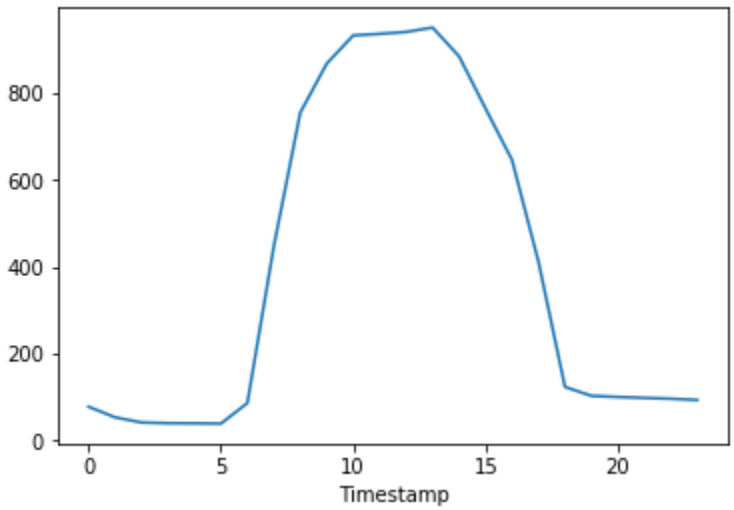
```
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(60,30))
for xcol, ax in zip(['temp', 'light','voltage'], axes):
    data.plot(kind='scatter', x='humidity', y=xcol, ax=ax, alpha=1, color='y'
```

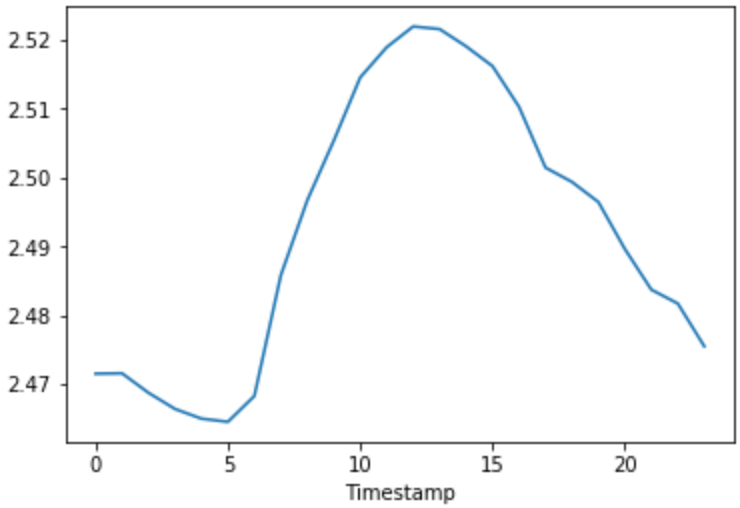
Variation in

Tempeature, Humidity, light and Voltage over time



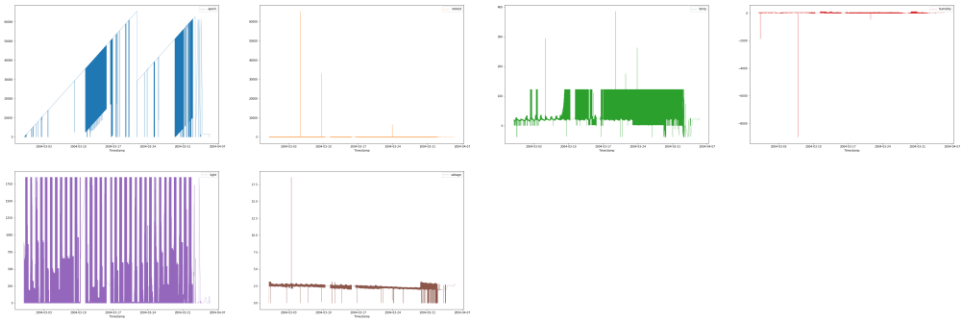






In [29]:

```
new_data.plot(subplots=True,linewidth=0.5,
                layout=(2, 4),figsize=(60, 20),
                sharex=False,
                sharey=False)

plt.show()
```



Pearson Correlation for the multivariate time series

In [30]:

```
new_data.corr(method='pearson')
```

Out[30]:

	epoch	moteid	temp	humidity	light	voltage
epoch	1.000000	0.006596	0.340746	-0.221278	0.041511	-0.654189
moteid	0.006596	1.000000	-0.014062	0.023541	0.029538	0.001789
temp	0.340746	-0.014062	1.000000	-0.707251	0.021948	-0.737583
humidity	-0.221278	0.023541	-0.707251	1.000000	-0.095084	0.507445
light	0.041511	0.029538	0.021948	-0.095084	1.000000	0.055835

voltage	-0.654189	0.001789	-0.737583	0.507445	0.055835	1.000000

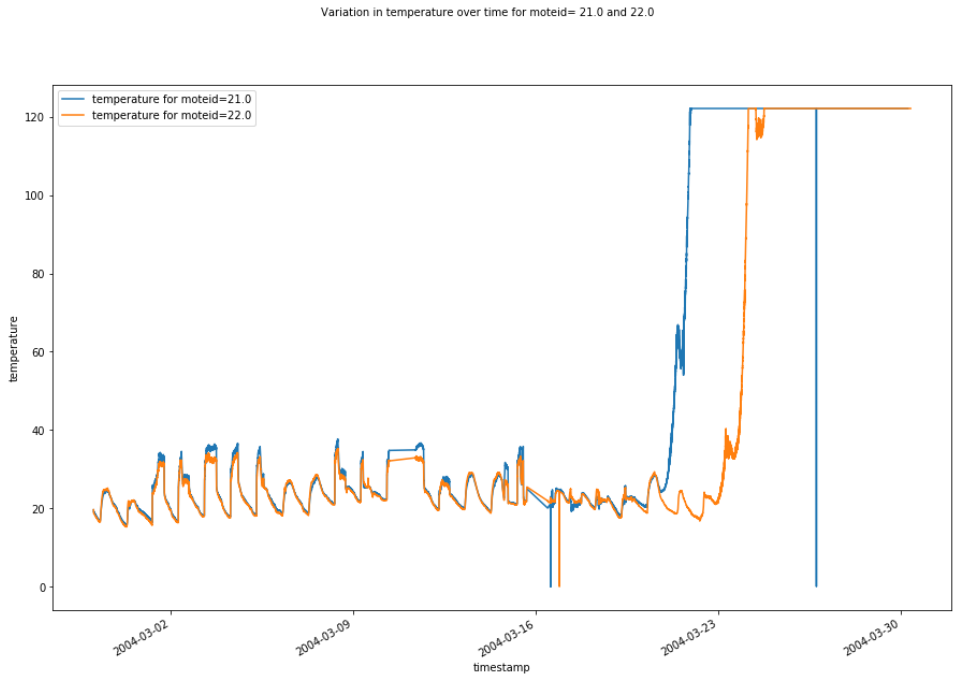
Variation in temperature readings over time for moteid's: 21 and 22

In [31]:

```
from matplotlib import pyplot as plt
d_m21 = data.loc[data['moteid'] == 21.0]
d_m22 = data.loc[data['moteid'] == 22.0]
d_m10 = data.loc[data['moteid'] == 10.0]
fig2 = plt.figure(figsize = (15,10))
d_m21['temp'].plot(label='temperature for moteid=21.0')
d_m22['temp'].plot(label='temperature for moteid=22.0')
fig2.suptitle('Variation in temperature over time for moteid= 21.0 and 22.0', fontsize=10)
plt.xlabel('timestamp', fontsize=10)
plt.ylabel('temperature', fontsize=10)
plt.legend()
```

Out[31]:

<matplotlib.legend.Legend at 0x7f8327b78ba8>



Anomaly Detection using moving average method

For moteid:10 and window size: 20, we calculate the mean and standard deviation of the data.If the next entry in the dataframe lies between mean(+-)sd*2, it is considered normal else it is considered an anamoly.

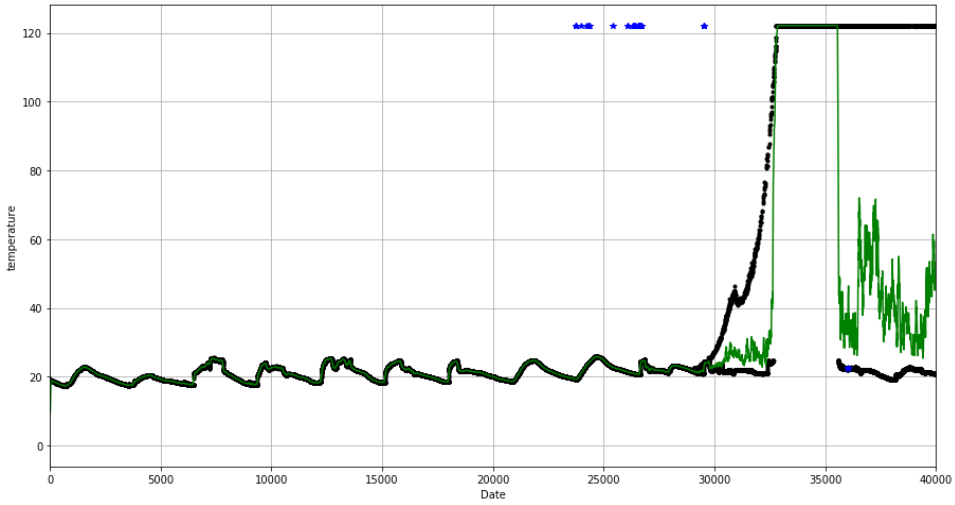
Anomaly can be seen by blue *

In [32]:

```
from itertools import count
import matplotlib.pyplot as plt
from numpy import linspace, loadtxt, ones, convolve
import numpy as np
import pandas as pd
import collections
from random import randint
from matplotlib import style
%matplotlib inline
def mov_average(data, window_size):

    window = np.ones(int(window_size))/float(window_size)
    return np.convolve(data, window, 'same')
def find_anomalies(y, window_size, sigma=1.0):
    avg = mov_average(y, window_size).tolist()
    residual = y - avg
    std = np.std(residual)
    return {'standard_deviation': round(std, 3),
            'anomalies_dict': collections.OrderedDict([(index, y_i) for index, y_i, avg_i
in zip(count(), y, avg)
              if (y_i > avg_i + (sigma*std)) | (y_i < avg_i - (sigma*std))])}
def plot_results(x, y, window_size, sigma_value=1,
                 text_xlabel="X Axis", text_ylabel="Y Axis", applying_rolling_std=False):

    plt.figure(figsize=(15, 8))
    plt.plot(x, y, "k.")
    y_av = mov_average(y, window_size)
    plt.plot(x, y_av, color='green')
    plt.xlim(0, 40000)
    plt.xlabel(text_xlabel)
    plt.ylabel(text_ylabel)
    events = {}
    events = find_anomalies(y, window_size=window_size, sigma=sigma_value)



    x_anom = np.fromiter(events['anomalies_dict'].keys(), dtype=int,
count=len(events['anomalies_dict']))
    y_anom = np.fromiter(events['anomalies_dict'].values(),
dtype=float,count=len(events['anomalies_dict']))
    plt.plot(x_anom, y_anom, "b*")
    print(x_anom)
    plt.grid(True)
    plt.show()
x = d_m10['epoch']
Y = d_m10['temp']
plot_results(x, y=Y, window_size=50, text_xlabel="Date",
sigma_value=3,text_ylabel="temperature")
```

```
[23743 23751 23761 23999 24206 24302 24303 24324 24350 25415 26094 26101
 26325 26336 26371 26422 26437 26549 26551 26581 26588 26622 26624 26636
 26713 26723 29505 29506 29507 36012 36017 40733 40748 40775 40779 40787
```

40840 40841 40848 40982 41115 41201]

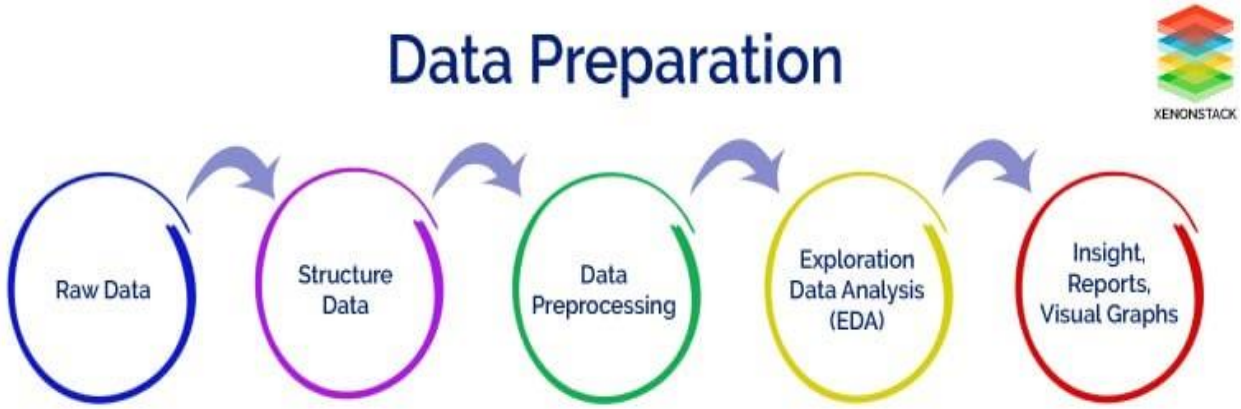


Introduction to Data Preparation

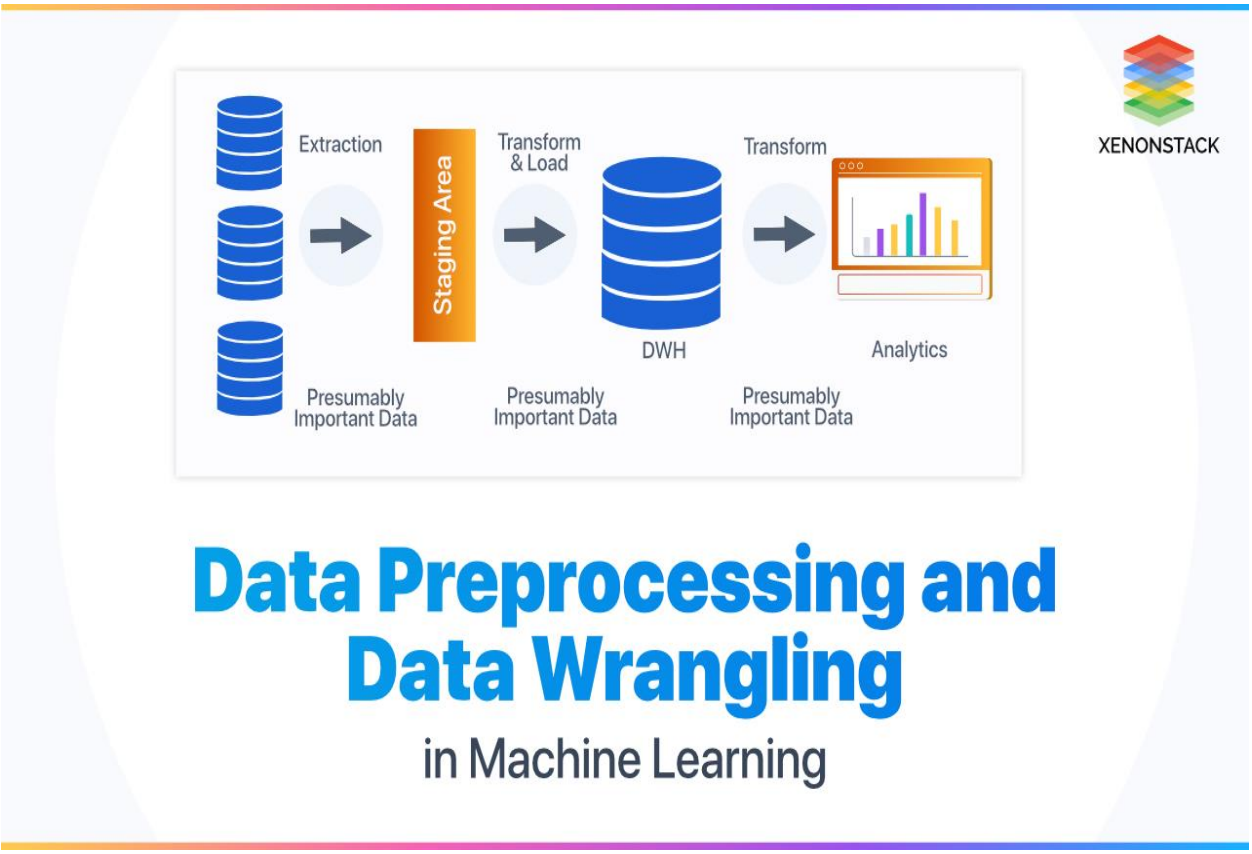
Deep learning and Machine learning are becoming more and more important in today's ERP (Enterprise Resource Planning). During the process of building the analytical model using Deep Learning or Machine Learning the data set is collected from various sources such as a file, database, sensors, and much more. But, the collected data cannot be used directly for performing the analysis process. Therefore, to solve this problem Data Preparation is done. It includes two techniques; Data Preprocessing and Data Wrangling

Data Preparation Architecture

Data Preparation process is an important part of Data Science. It includes two concepts such as Data Cleaning and Feature Engineering. These two are compulsory for achieving better accuracy and performance in the Machine Learning and Deep Learning projects.



For achieving better results from the applied model in Machine Learning and Deep Learning projects the format of the data has to be in a proper manner, this is where term Data Preparation is used. Some specified Machine Learning and Deep Learning model need information in a specified format, for example, Random Forest algorithm does not support null values, therefore to execute random forest algorithm null values has to be managed from the original raw data set. Another aspect of Data Preparation and analysis is that the data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithms are executed in one data set, and the best out of them is chosen.



It is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. Therefore, certain steps are executed to convert the data into a small clean data set. This technique is performed before the execution of the Iterative Analysis. The set of steps is known as Data Preprocessing. It includes -

- Data Cleaning
- Data Integration
- Data Transformation
- Data Reduction

A product of Apache Software Foundation, which is in an open-source unified programming model and is used to define and execute data processing pipelines.

Preprocessing of Data is necessary because of the presence of unformatted real-world data (Raw Data). Mostly real-world data is composed of -

- **Inaccurate data (missing data) -** There are many reasons for missing data such as data is not continuously collected, a mistake in data entry, technical problems with biometrics, and much more, which requires proper Data Preparation.
- **The presence of noisy data (erroneous data and outliers) -** The reasons for the existence of noisy data could be a technological problem of gadget that gathers data, a human mistake during data entry and much more.
- **Inconsistent data -** The presence of inconsistencies are due to the reasons such that existence of duplication within data, human data entry, containing mistakes in codes or names, i.e., violation of data constraints and much more necessitate Data Preparation and analysis.

Data Wrangling is an important aspect of implementing the model. Therefore, data is converted to the proper feasible format before applying any model to it. By performing filtering, grouping, and selecting appropriate data accuracy and performance of the model could be increased. Another concept is that when time-series data has to be handled every algorithm is executed with different aspects. Therefore it is used to convert the time series data into the required format of the applied model. In simple words, the complex data is transformed into a usable format for performing analysis on it.

We spend 80% of our efforts on data preparation, not on analysis and execution.

It is used to handle the issue of **Data Leakage** while implementing Machine Learning and Deep Learning. First of all, we have to understand what Data Leakage is?

Data Leakage in Machine Learning and Deep Learning

Data Leakage is responsible for the cause of an invalid Machine Learning/Deep Learning model due to the over-optimization of the applied model. Data Leakage is the term used when the data from outside, i.e., not part of the training dataset is used for the learning process of the model. This additional learning of information by the applied model will disapprove of the computed estimated performance of the model. For example when we want to use the particular feature for performing **Predictive Analysis**, but that specific feature is not present at the time of training of dataset then data leakage will be introduced within the model. Data Leakage can be demonstrated in many ways that are given below -

- The Leakage of data from test dataset to the training data set.
- Leakage of computed correct prediction to the training dataset.
- Leakage of future data into the past data.
- Usage of data outside the scope of the applied algorithm

In general, the leakage of data is observed from two primary sources of Machine Learning/Deep Learning algorithms such as feature attributes (variables) and training data set.

Checking the presence of Data Leakage within the applied model

Data Leakage is observed at the time of usage of complex datasets. They are described below -

- At the time of dividing the time series dataset into training and test, the dataset is a complex problem.
- The implementation of sampling in a graphical problem is a complex task.
- Storage of analog observations in the form of audios and images in separate files having a defined size and timestamp.

Data centers are designed and deployed to provide storage for critical data and some applications of the organizations.

Data Preprocessing is carried out to remove the cause of unformatted real-world data which we discussed above. First of all, let's explain how missing data can be handled during Data Preparation. Three different steps can be executed which are given below -

- **Ignoring the missing record -** It is the simplest and efficient method for handling the missing data. But, this method should not be performed at the time when the number of missing values is immense or when the pattern of data is related to the unrecognized primary root of the cause of the statement problem.
- **Filling the missing values manually -** This is one of the best-chosen methods of Data Preparation process. But there is one limitation that when there are large data set, and missing values are significant then, this approach is not efficient as it becomes a time-consuming task.
- **Filling using computed values -** The missing values can also be occupied by computing mean, mode or median of the observed given values. Another method could be the predictive values in Preprocessing of Data is that are computed by using any Machine Learning or Deep Learning tools and algorithms. But one drawback of this approach is that it can generate bias within the data as the calculated values are not accurate concerning the observed values.
- Let's move further and discuss how we can deal with noisy data. These are the famous methods that can be followed for Data Preprocessing and analysis -

- Data Binning
- Preprocessing in Clustering
- Machine Learning
- Removing manually

Data Binning

Data binning, or data bucketing, is a data pre-processing procedure used to reduce the effects of little observation errors. The actual data values which fall into a given small interval, a bin, are replaced by a value representative of that interval, often mean or median. This method is also known as local smoothing. There are two types of binning:

- **Unsupervised Binning**: Equal width binning, Equal frequency binning.
- **Supervised Binning**: Entropy-based binning

Preprocessing in Clustering

In the approach, the outliers may be detected by grouping similar data in the same group, i.e., in the same cluster.

Machine Learning

A Machine Learning algorithm can be executed for the smoothing of data during Preprocessing . For example, Regression Algorithm can be used for the smoothing of data using a specified linear function.

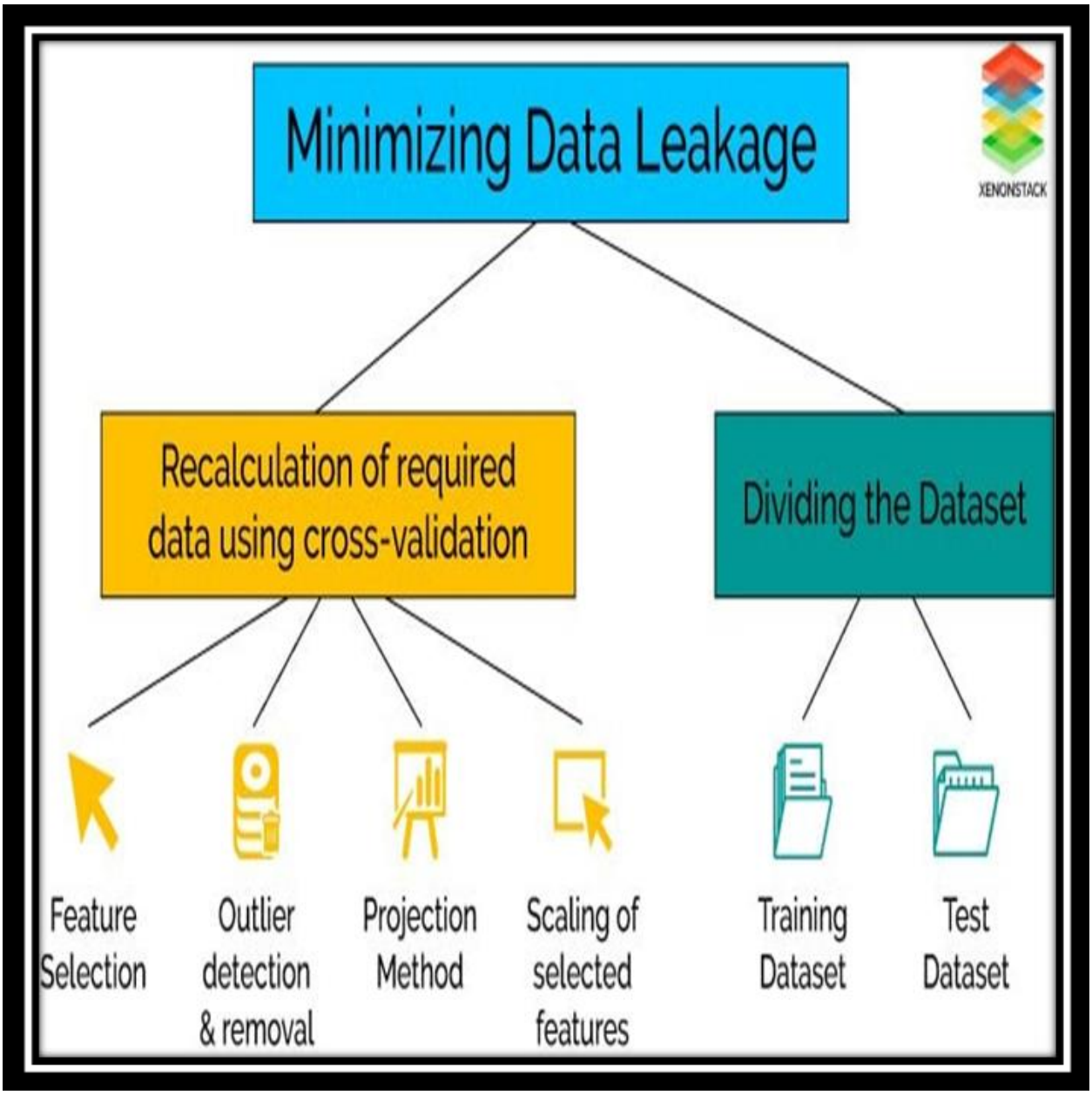
Removing manually

The noisy data can be deleted manually by the human being, but it is a time-consuming Data Preparation process, so mostly this method is not given priority. To deal with the **inconsistent data** manually and perform Data Preparation and analysis properly, the data is managed using external references and knowledge engineering tools like the knowledge engineering process.

Data Wrangling is conducted to minimize the effect of Data Leakage while executing the model. Suppose one considers the complete data set for normalization and standardization. In that case, cross-validation is performed to estimate the model's performance leads to the beginning of data leakage. Another problem is observed in that the test model is also included for feature selection while executing each cross-validation fold, which further generates bias during performance

analysis. Data Leakage's effect is minimized by performing Data Preparation during the cross-validation process.

The cross-Validation process includes feature selection, outlier detection and removal, projection methods, scaling of selected features, and more. Another solution for better Data Preparation is dividing the complete dataset into a training data set used to train the model and a validation dataset used to evaluate the applied model's performance and accuracy. The model selection is made by looking at the results of the test data set in the cross-validation process. This conclusion will not always be valid as the sample of the test data set could vary. The performance of different models is evaluated for the particular test dataset. Therefore, while selecting the best model, test error is overfitting. The test error variance is determined by using different samples of the test dataset. Choosing of suitable model happens in this way.



Data Preprocessing steps are performed before the Wrangling. In this case, data is prepared exactly after receiving the data from the data source. In this initial transformations, Data Cleaning or any aggregation of data is performed. It is executed once. For example, we have data where one attribute has three variables, and we have to convert them into three attributes and delete the special characters from them. The concept of Data Preparation steps performed before applying any iterative model and will be executed once in the project. On the other hand, Wrangling is performed during the iterative analysis and model building. This concept at the time of feature engineering. The conceptual view of the dataset changes as different models are applied to achieve a good analytic model.

For example, we have data containing 30 attributes where two attributes are used to compute another attribute, and that computed feature is used for further analysis. In this way, the data

could be changed according to the requirement of the applied model, and Data Preparation can be effective.

Tasks of Data Preparation

Different steps are involved in Data Preprocessing. These steps are described below:

- Data Cleaning
- Data Integration
- Data Transformation
- Data Reduction

Data Cleaning

This is the first step which is implemented in Preprocessing. In this step, the primary focus is on handling missing data, noisy data, detection, and removal of outliers minimizing duplication, and computed biases within the data.

Data Integration

This process is used when data is gathered from various data sources and data are combined to form consistent data. This consistent data after performing data cleaning is used for Data Preparation and analysis.

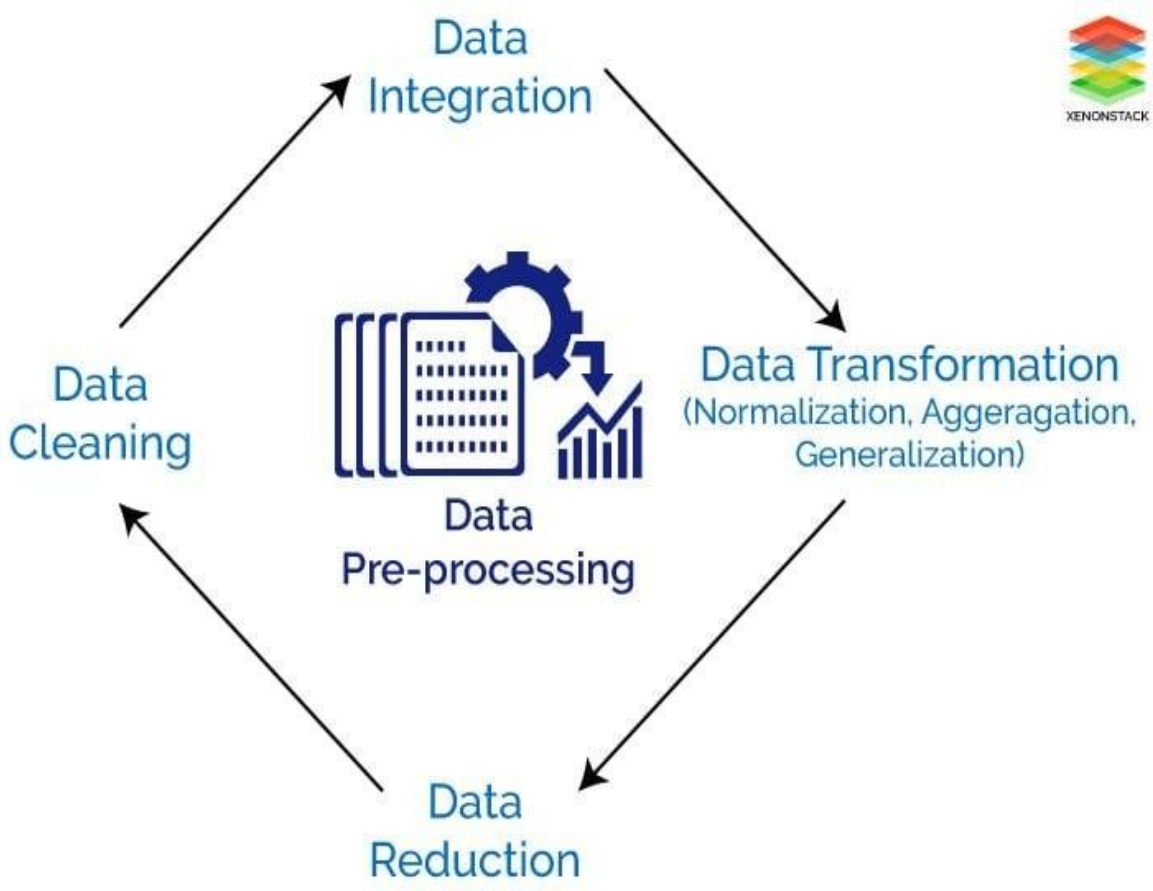
Data Transformation

This step is used to convert the raw data into a specified format according to the need of the model. The options for the transformation of data are given below -

- **Normalization -** In this method, numerical data is converted into the specified range, i.e., between 0 and one so that scaling of data can be performed.
- **Aggregation -** The concept can be derived from the word itself, this method is used to combine the features into one. For example, combining two categories can be used to form a new group.
- **Generalization -** In this case, lower level attributes are converted to a higher standard.

Data Reduction

After the transformation and scaling of data duplication, i.e., redundancy within the data is removed and efficiently organize the data during Data Preparation.



The tasks of Data wrangling are described below -

- Discovering
- Structuring
- Cleaning
- Enrichment
- Validating
- Publishing

Discovering

Firstly, data should be understood thoroughly and examine which approach will best suit. For example: if have weather data when we analyze the data it is observed that data is from one area and so primary focus is on determining patterns.

Structuring

As the data is gathered from different sources, the data will be present in various shapes and sizes. Therefore, there is a need for structuring the data in a proper format.

Cleaning

Cleaning or removing of data should be performed that can degrade the performance of the analysis.

Enrichment

Extract new features or data from the given data set to optimize the performance of the applied model.

Validating

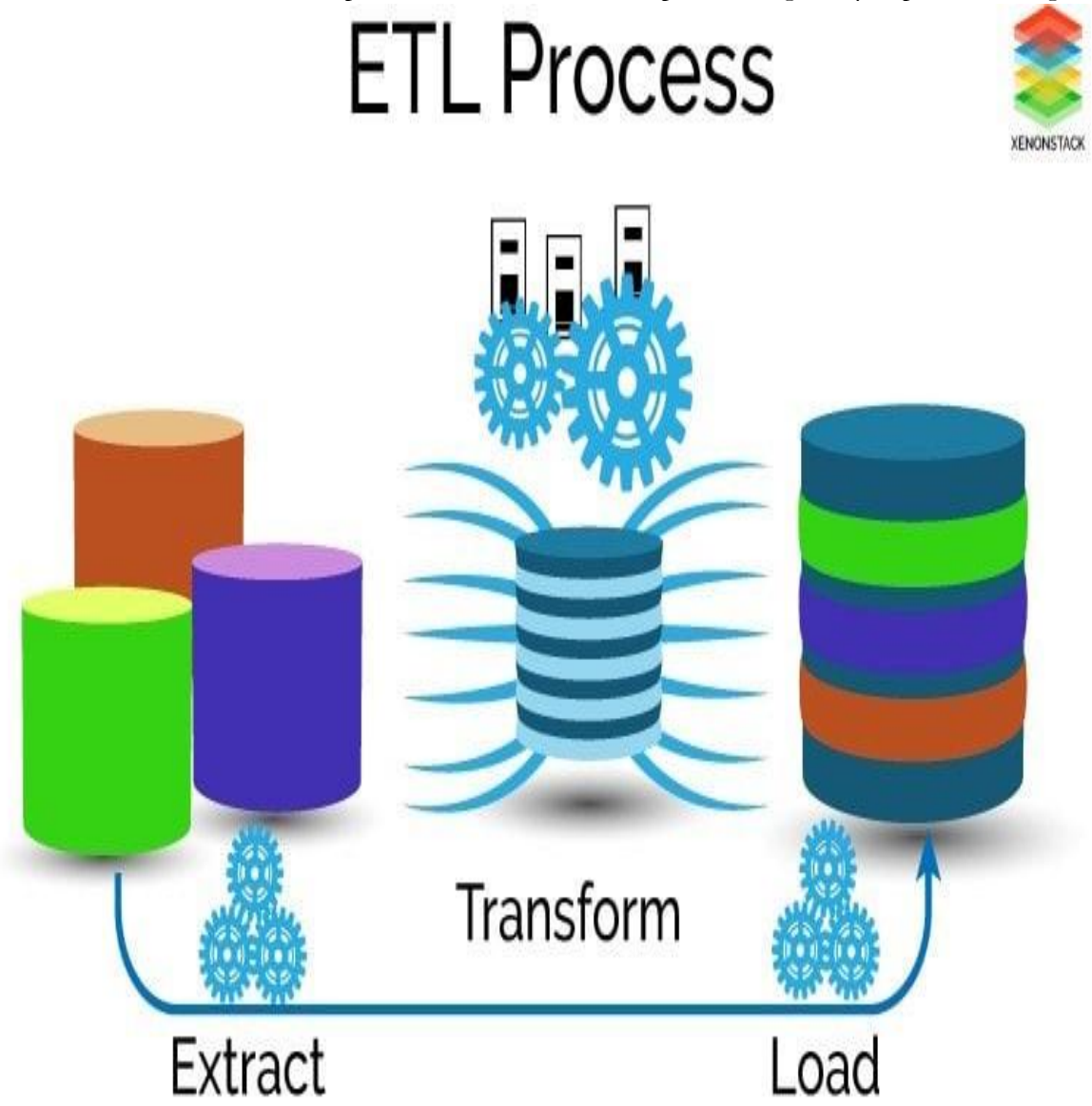
This approach is used for improving the quality of data and consistency rules so that transformations that are applied to the data could be verified.

Publishing

After completing the steps of Data Wrangling, the steps can be documented so that similar steps can be performed for the same kind of data to save time.

With the advancement in the technology and generation of data, data is collected from various sources. Therefore, in the Data Preparation process managing data in different formats is necessary. As the simple Data Preparation and analysis methods alone are not feasible for the complex problem statement, it is introduced which simplifies the analysis process of a complex issue. In this way, Data Wrangling is used for improving the analysis process of complex problems during Data Preparation.

Wrangling technology is used by business analysts, users engaged in business, and managers. On the other hand, ETL (Extract, Transform, and Load) is employed by IT Professionals. They receive the requirements from business people and then they use ETL tools to deliver the data in a required format. Data Wrangling is used to analyze the data that was gathered from different data sources. It is designed specially to handle diverse and complex data of any scale. But in the case of ETL, it can handle structured data that was originated from different databases or operating systems. The primary task of the Wrangling method is to manage the newly generated data from various sources for the analysis process whereas the goal of ETL is to extract, transform and load the data into the central enterprise Data Warehouse for performing analysis process using b



- **R:** R a framework that consists of various packages that can be used for Data Preprocessing like dplyr etc.

- **Weka:** Weka is a software that contains a collection of Machine Learning algorithms for the Data Mining process. It consists of Preprocessing tools that are used before applying Machine Learning algorithms.
- **RapidMiner:** RapidMiner is an open-source Predictive Analytics Platform for Data Mining process. It provides efficient tools for performing the exact Data Preprocessing process.
- **Python:** Python is a programming language that provides various libraries that are used for Preprocessing.

- **Tabula:** Tabula is a tool that is used to convert the tabular data present in pdf into a structured form of data, i.e., spreadsheet.
- **OpenRefine:** OpenRefine is open-source software that provides a friendly Graphical User Interface (GUI) that helps to manipulate the data according to your problem statement and makes Data Preparation process simpler. Therefore, it is highly useful software for the non-data scientist.
- **R:** R is an important programming language for the data scientist. It provides various packages like dplyr, tidyr, etc. for performing data manipulation.
- **Data Wrangler:** Data Wrangler is a tool that is used to convert real-world data into the structured format. After the conversion, the file can be imported into the required application like Excel, R, etc. Therefore, less time will be spent on formatting data manually.
- **CSVKit:** CSVKit is a toolkit that provides the facility of conversion of CSV files into different formats like CSV to JSON, JSON to CSV, and much more. It makes the process of wrangling easy.
- **Python with Pandas:** Python is a language with Pandas library. This library helps the data scientist to deal with complex problems efficiently and makes Data Preparation process efficient.