SyedHassaanTauqeer 19-01-2019 Final Code Notebook Draft

In [4]:
```
import os, glob
import pandas as pd
import numpy as np
```

In [18]:
```
def file_write(df, path, name):
    #This is a simple function that writes a dataframe to a csv file
    return df.to_csv(path+name+'.csv', sep = ';', index = False)#Making index
 False helps ignore index column when writing file
```

In [21]:
```
def floatConv(arr):#This function takes in a python list/ numpy array and repl
aces any commas left in the thousand position in
    #numerical values. This stands as a safeguard to ensure all values are flo
at for easier future operations
    temp = []
    for i in range(len(arr)):
        temp.append( float(str(arr[i]).replace(",", "") ) )
    return temp
```

In [22]:
```
def perScale(arr):#This function is used to scale a list or numpy array to per
centage ratio.
    #This is done by taking the maximum value in that array and scaling every
 other value relatively. No trailing decimal places
    #left for the sake of ease in matching and querying at future stages
    scaled = []
    tempMax = max(arr)
    for i in range(len(arr)):
        scaled.append( round(float(arr[i]/tempMax)*100, 0) )
    return scaled
```

In [23]:
```
def minFil(arrList, win):#This function takes in a list/numpy array and applie
s a rolling window operation to get the minimum
    #value within that window.
    tempDiff = arrList
    appVal = arrList[len(arrList)-1]# To ensure the input and output lengths o
f the array are same, the last value of the input
    #array is repeated the number of times that matches one less than the wind
ow size
    tempDiff = np.insert(tempDiff, len(tempDiff), np.repeat(appVal, win-1))

    arrListDF = pd.DataFrame({"arr":tempDiff})#since the rolling function in p
andas is really efficient, we convert the array
    #to Series data
    tempDiff = arrListDF.rolling(win).min().dropna()#all NaN values are droppe
d to not cause issues in future operations
    tempDiff = tempDiff['arr'].values

    return tempDiff
```

In [24]:
```
def diffIndexMaker(arr):#This function takes in an numpy array/list and comput
es the difference in between each consecutive value
    #To ensure that the input and output lengths of the array remain same, the
 first value of the array is pre-pended.
    #This doesn't affect the computation or any future operations but keeps le
ngth consistent
    temp = arr.item(0)
    arr = np.insert(arr, 0, temp)#pre-pending the first value to the array
    return np.diff(arr)
```

This block makes all the data files uniform in shape

The data that is re-structured by the following block is then taken to label the "spin phases" and the "end spin phases". The labeled files are copies of this format

In [42]:
```
#Change working directory to raw data folder
os.chdir('C:\\Users\\Labyrinth\\JUPYTER NOTEBOOKS\\WeWash_Praktikum_TUM3sem\\W
eWash_Analysis_ver2\\Data\\raw')
```

In [43]:
```
for _file in glob.glob("*.csv"): #Iterating through all files in the raw data
 directory
    writingDF = pd.DataFrame(columns=['machine', 'pow', 'time', 'timeStamp'])
    labtsDF = pd.read_csv(_file, delimiter=',')

    machines = labtsDF['reservation_id'].unique() #Uniques machines are filter
ed based on id
    print 'machines: ', len(machines)
    for i in range(len(machines)):#Each of the machines has multiple records.
        tempDF = labtsDF[labtsDF['reservation_id'] == machines[i]]
        subSet = tempDF[['reservation_id', 'power', 'sample_time']] #Subset th
e dataframe on relevant columns
        subSet['time'] = np.arange(len(subSet['power'].values))# Making time s
tamps uniform. This is just an incremental order
        #given to timestamps. e.d 0,1,2,3,4 instead of UNIX timestamps which c
ompromises uniformity
        subSet.rename(index=str, columns={"reservation_id": "machine", "power"
: "pow", "sample_time":"timeStamp"}, inplace=True)
        #The headers are re-named to match the writing data frame for seamless
 stitching of records
        writingDF = writingDF.append(subSet)

    savingPath = 'C:\\Users\\Labyrinth\\JUPYTER NOTEBOOKS\\WeWash_Praktikum_TU
M3sem\\WeWash_Analysis_ver2\\Data\\intermediate\\'
    fileName = 'uniform_'+str(len(machines))+'_'+str(machines[0])+'-'+str(mach
ines[len(machines)-1])
    file_write(writingDF, path= savingPath, name= fileName)
```

```
machines:  50

C:\Users\Labyrinth\Anaconda\lib\site-packages\ipykernel_launcher.py:10: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  # Remove the CWD from sys.path while we load stuff.

machines:  150
machines:  146
machines:  400
```

This part prepares the Labelled data for the transformation to the training format

In [45]:
```
#Change working directory to labelled training data folder
os.chdir('C:\\Users\\Labyrinth\\JUPYTER NOTEBOOKS\\WeWash_Praktikum_TUM3sem\\W
eWash_Analysis_ver2\\Data\\intermediate\\labelled\\train')
```

In [48]:
```
bigDF = pd.DataFrame() #This is the master dataframe which will contain all th
e training data. In this case we shall have
#data for about 146+150+400 = 900 Machines. Individual records are definitely
 way more than that.
for _file in glob.glob("*.csv"): #Iterating through all files in the directory
    labtsDF = pd.read_csv(_file, delimiter=';')
    machines = labtsDF['machine'].unique() #Uniques machines are filtered base
d on id
    print 'machines: ', len(machines)

    for i in range(len(machines)):#Each of the machines has multiple records.
        tempDF = labtsDF[labtsDF['machine'] == machines[i]]#now that we have a
 subset of the dataframe filtered on
        #the machine, we'll add features to it and then append it to the main
 dataframe.
        tempDF['pow'] = floatConv(tempDF['pow'].values)#float conversions
        tempDF['time'] = floatConv(tempDF['time'].values)#float conversions
        tempDF['scaledPower'] = perScale(tempDF['pow'].values)#percentage scal
ing
        tempDF['scaledTime'] =  perScale(tempDF['time'].values)#percentage sca
ling
        tempDF['powMin'] = minFil(tempDF['pow'].values, win=5)#applying minimu
m filter to main power values
        tempDF['scaledPowMin'] = minFil(tempDF['scaledPower'].values, win=5)#a
pplying minimum filter to scaled power values
        tempDF['scaledPowDiff'] = diffIndexMaker(tempDF['scaledPower'].values)
#creating the difference index from scaled power
        bigDF = bigDF.append(tempDF)

print len(bigDF['machine'].unique())
print bigDF.head()
```

```
machines:  118

C:\Users\Labyrinth\Anaconda\lib\site-packages\ipykernel_launcher.py:11: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  # This is added back by InteractiveShellApp.init_path()
C:\Users\Labyrinth\Anaconda\lib\site-packages\ipykernel_launcher.py:12: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  if sys.path[0] == '':
C:\Users\Labyrinth\Anaconda\lib\site-packages\ipykernel_launcher.py:13: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  del sys.path[0]
C:\Users\Labyrinth\Anaconda\lib\site-packages\ipykernel_launcher.py:14: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy

C:\Users\Labyrinth\Anaconda\lib\site-packages\ipykernel_launcher.py:15: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  from ipykernel import kernelapp as app
C:\Users\Labyrinth\Anaconda\lib\site-packages\ipykernel_launcher.py:16: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  app.launch_new_instance()
C:\Users\Labyrinth\Anaconda\lib\site-packages\ipykernel_launcher.py:17: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
```

```
machines:  146
machines:  150
machines:  282
696
   end_spin  machine    pow  powMin  scaledPowDiff  scaledPowMin  scaledPower
  \
0       NaN   211286  0.094   0.086            0.0           0.0          0.0

1       NaN   211286  0.090   0.082            0.0           0.0          0.0

2       NaN   211286  0.086   0.082            0.0           0.0          0.0

3       NaN   211286  0.086   0.082            0.0           0.0          0.0

4       NaN   211286  0.086   0.074            0.0           0.0          0.0


   scaledTime  spins  time
0         0.0    NaN   0.0
1         0.0    NaN   1.0
2         0.0    NaN   2.0
3         0.0    NaN   3.0
4         1.0    NaN   4.0
```

In [49]:
```
#It is extremely important to fill all NaN values because otherwise it will ha
mper the training process
bigDF['end_spin'] = bigDF['end_spin'].fillna(0.0)
bigDF['spins'] = bigDF['spins'].fillna(0.0)
bigDF['pow'] = bigDF['pow'].fillna(0.0)
```

In [50]:
```
#Time to finally write the training dataframe to a csv file
savingPath = 'C:\\Users\\Labyrinth\\JUPYTER NOTEBOOKS\\WeWash_Praktikum_TUM3se
m\\WeWash_Analysis_ver2\\Data\\processed\\'
fileName = 'UniFeatScaledV2_146-150-400_696_128895-212236'
file_write(bigDF, path=savingPath, name=fileName)
```