DISCnet Placement at APSensing UK

-DAS Prognostic Health Monitoring with Machine Learning-

Arran Freegard

Supervisors: U. Blumenschein, A. Belyaev & S. Moretti

APSensing UK

- HP (1939) and Agilent Technologies (1999) heritage
- German based optical sensing technologies distributor (for over 25 years):
  - Distributed Temperature Sensors (DTS)
  - Distributed Acoustic Sensors (DAS)
  - Distributed Thermal Gradient Sensors (DTGS)
- Subsea and land cables with DAS and DTS power cable monitoring systems
- UK based Data Science Office in Basingstoke, focusing on Alarm systems using Machine Learning techniques

Project I: Telemetry Dashboards with ML predictions

- Analyse time series data produced by DAS systems to monitor hardware health

- Parameters recorded and sent to a database periodically:
  - Disk and internal Temperatures
  - Fan Speeds
  - Disk Usages
  - Status/Error codes
  - Uptime/Reboot cycles

- Build a predictive model to determine current operating health status and future failure

Telemetry Dashboard #1: Variables

InternalTemp1

diskTemp1,2,3

diskUsage1,2,3

Fan1,2

Adds days if reboot cycle is constant

DAS Systems with Telemetry

Status			Date / Time	Serial Number	Uptime	InternalTemp (degC)		Disk Temp (degC)				Total Disk Usage (TB)			Fan Speed (RPM)		Machine Learning Verdict				Reboot Cycles	Days Booted	Total days booted	Version	Update Date
DAS	IU	PU				IU	PU	nvme	sda	sdb	sdc	sda	sdb	sdc	Fan1	Fan2	Linear Regressions	SVM	IsolationForest	Autoencoder					
✓	1	LIVE	16/12/19 23:30	DE5000PP02	1w 14h 36m	52	48	48	44	45	44	918.7	918.5	53.2	5900	5900	Normal	Normal	Normal	Normal	579	167	231	1.1.4.3643	09/12/2019
!	1	Standby	16/12/19 23:30	DE52000218	4d 13h 10m	50	51	45	44	44	44	8.6	8.6	1.7	6200	6100	Normal	Normal	Normal	Normal	25	4	74	1.1.4.3644	09/12/2019
✓	1	LIVE	16/12/19 23:30	DE52000235	3w 3d 14h 48m	45	52	52	46	46	46	10.7	10.7	0.4	6800	6800	Normal	ANOMALY	Normal	Normal	27	24	30	1.1.4.3526	03/12/2019
✓	1	LIVE	16/12/19 23:30	DE52000250	6w 5d 9h 48m	33	44	40	39	39	38	8.6	8.6	1.9	5100	5200	Normal	Normal	Normal	Normal	19	46	117	1.1.4.3213	12/11/2019
✓	1	LIVE	16/12/19 23:30	DE52000LP2	1w 4d 16h 10m	37	39	46	44	44	42	398.6	398.5	54.7	5800	5900	Normal	Normal	Normal	Normal	58	11	149	1.1.3.3094	19/11/2019
!	0	-NOT	16/12/19 23:30	DE5200DV01	7w 4d 12h 34m	0	43	35	36	36	36	6.1	6.1	0.1	4900	4900	Normal	ANOMALY	Normal	Normal	13	200	200	1.1.4.3423	25/11/2019
!	0	-N(	15/12/19 23:36	DE52000LP1	2d 9h 12m	ERROR	42	40	36	37	37	277.5	277.5	277.5	5200	5100	FanRMSE: +/- 1759.33	Normal	ANOMALY	ANOMALY	6	2	188	1.1.3.3094	11/11/2019
!	1	OK	04/12/19 11:33	DE52000219	34m	0	28	24	26	31	28	65.4	58.1	61.9	3800	3600	nvme Temp RMSE: +/- 14755.6	Normal	ANOMALY	ANOMALY	75	0	32	1.1.0	21/06/2019
!	1	OK	28/11/19 11:30	DE52000LP3	6d 17h 42m	46	41	40	37	38	37	0.0	0.0	0.0	4100	3900	Normal	Normal	ANOMALY	Normal	64	12	177	1.1.4.3423	27/11/2019
!	1	OK	24/11/19 23:30	DE52000253	1w 3d 6h 35m	37	32	34	28	28	29	0.0	0.0	0.0	3900	3800	Normal	Normal	ANOMALY	ANOMALY	28	10	10	1.1.3.3094	21/11/2019
!	1	OK	15/09/19 23:31	DE5200PP04	6d 12h 35m	42	40	44	42	42	40	175.9	175.9	9.7	5100	5000	Normal	Normal	ANOMALY	Normal	104	6	82	1.1.1.2584	30/08/2019
!	0	ive: -IU NOT r	19/08/19 23:30	DE52000PP1	5d 9h 23m	0	45	40	41	41	40	8.0	8.0	1.9	5400	5500	FanRMSE: +/- 3552.727	Normal	ANOMALY	Normal	18	5	26	1.1.1.2245	24/07/2019
!	1	-NO	23/07/19 23:30	DE52000201	5w 5d 6h 41m	ERROR	46	40	41	42	41	8.0	8.0	1.9	5300	5400	Normal	Normal	ANOMALY	Normal	16	39	105	1.1.0	11/06/2019
!	1	-NO	03/07/19 23:31	DE52000229	12h 31m	ERROR	45	39	41	42	41	0.7	0.7	0.1	4000	4100	Normal	Normal	ANOMALY	Normal	29	0	0	1.1.0	04/06/2019
!	0	IU NOT ready	29/05/19 12:50	DE5xxxxxxx	4w 5d 1h 59m	0	38	28	34	35	34	1.1	1.1	0.1	4400	4400	Normal	Normal	ANOMALY	Normal	0	75	87	1.0.10	0.0
!	1	-NO	06/02/19 23:30	DE5200PP01	7h 43m	ERROR	31	30	38	38	36	60.5	60.5	4.0	4300	4300	FanRMSE: +/- 126079.473	Normal	ANOMALY	Normal	38	0	6	1.0.8RC3	Wed 6 Feb 2019

Machine learning predictions based on different models

Telemetry Dashboard #1: Manual Thresholds

Temperatures ≥ 50 degrees

Disk Usage ≥ 700 TB

Fan Speed ≥ 6000 RPM

Days Booted < 1 (reboot cycle has counted)

Live

NOT live

Status/Error code from 'TBD12' (defined on wiki)

Not sent in >1 day

Red text: Has an error

IU not present, but files left over

AP SENSING advanced photonic

DAS Systems with Telemetry

Status			Date / Time	Serial Number	Uptime	Internal Temp (degC)		Disk Temp (degC)				Total Disk Usage (TB)			Fan Speed (RPM)		Machine Learning Verdict				Reboot Cycles	Days Booted	Total days booted	Version	Update Date
DAS	IU	PU				IU	PU	nvme	sda	sdb	sdc	sda	sdb	sdc	Fan1	Fan2	Linear Regressions	SVM	IsolationForest	Autoencoder					
✓	1	LIVE	16/12/19 23:30	DE5000PP02	1w 14h 36m	52	48	48	44	45	44	918.7	918.5	53.2	5900	5900	Normal	Normal	Normal	Normal	579	167	231	1.1.4.3643	09/12/2019
!	1	Standby	16/12/19 23:30	DE52000218	4d 13h 10m	50	51	45	44	44	44	8.6	8.6	1.7	6200	6100	Normal	Normal	Normal	Normal	25	4	74	1.1.4.3644	09/12/2019
✓	1	LIVE	16/12/19 23:30	DE52000235	3w 3d 14h 48m	45	52	52	46	46	46	10.7	10.7	0.4	6800	6800	Normal	ANOMALY	Normal	Normal	27	24	30	1.1.4.3526	03/12/2019
✓	1	LIVE	16/12/19 23:30	DE52000250	6w 5d 9h 48m	33	44	40	39	39	38	8.6	8.6	1.9	5100	5200	Normal	Normal	Normal	Normal	19	46	117	1.1.4.3213	12/11/2019
✓	1	LIVE	16/12/19 23:30	DE52000LP2	1w 4d 16h 10m	37	39	46	44	44	42	398.6	398.5	54.7	5800	5900	Normal	Normal	Normal	Normal	58	11	149	1.1.3.3094	19/11/2019
!	0	-NOT	16/12/19 23:30	DE5200DV01	7w 4d 12h 34m	0	43	35	36	36	36	6.1	6.1	0.1	4900	4900	Normal	ANOMALY	Normal	Normal	13	200	200	1.1.4.3423	25/11/2019
!	0	-N(	15/12/19 23:36	DE52000LP1	2d 9h 12m	ERROR	42	40	36	37	37	277.5	277.5	277.5	5200	5100	FanRMSE: +/- 1759.33	Normal	ANOMALY	ANOMALY	6	2	188	1.1.3.3094	11/11/2019
!	1	OK	04/12/19 11:33	DE52000219	34m	0	28	24	26	31	28	65.4	58.1	61.9	3800	3600	nvme Temp RMSE: +/- 14755.6	Normal	ANOMALY	ANOMALY	75	0	32	1.1.0	21/06/2019
!	1	OK	28/11/19 11:30	DE52000LP3	6d 17h 42m	46	41	40	37	38	37	0.0	0.0	0.0	4100	3900	Normal	Normal	ANOMALY	Normal	64	12	177	1.1.4.3423	27/11/2019
!	1	OK	24/11/19 23:30	DE52000253	1w 3d 6h 35m	37	32	34	28	28	29	0.0	0.0	0.0	3900	3800	Normal	Normal	ANOMALY	ANOMALY	28	10	10	1.1.3.3094	21/11/2019
!	1	OK	15/09/19 23:31	DE5200PP04	6d 12h 35m	42	40	44	42	42	40	175.9	175.9	9.7	5100	5000	Normal	Normal	ANOMALY	Normal	104	6	82	1.1.1.2584	30/08/2019
!	0	ive: -IU NOT r	19/08/19 23:30	DE52000PP1	5d 9h 23m	0	45	40	41	41	40	8.0	8.0	1.9	5400	5500	FanRMSE: +/- 3552.727	Normal	ANOMALY	Normal	18	5	26	1.1.1.2245	24/07/2019
!	1	-NO	23/07/19 23:30	DE52000201	5w 5d 6h 41m	ERROR	46	40	41	42	41	8.0	8.0	1.9	5300	5400	Normal	Normal	ANOMALY	Normal	16	39	105	1.1.0	11/06/2019
!	1	-NO	03/07/19 23:31	DE52000229	12h 31m	ERROR	45	39	41	42	41	0.7	0.7	0.1	4000	4100	Normal	Normal	ANOMALY	Normal	29	0	0	1.1.0	04/06/2019
!	0	IU NOT ready	29/05/19 12:50	DE5xxxxxxx	4w 5d 1h 59m	0	38	28	34	35	34	1.1	1.1	0.1	4400	4400	Normal	Normal	ANOMALY	Normal	0	75	87	1.0.10	0.0
!	1	-NO	06/02/19 23:30	DE5200PP01	7h 43m	ERROR	31	30	38	38	36	60.5	60.5	4.0	4300	4300	FanRMSE: +/- 1260793.473	Normal	ANOMALY	Normal	38	0	6	1.0.8RC3	Wed 6 Feb 2019

Telemetry Dashboard #1: Machine Learning Columns

ML Process:

- Trim data to remove anomalies for training:
  - Certain TBD12 codes
  - No command timeouts
  - Systems that have been running for more than a day
  - Temperatures less than 60
  - Fan speeds less than 7000
  - Removed data 12 hours before shut downs
- Trains (and validate) models on this healthy data set
- diskUsage1, diskUsage3, diskTemp1, Fan1, internalTemp1, TBD2 (PUTemp), uptime, BootTimer, Live (based on VIF analysis)
- NOT trained per ID (this gave bad results, not enough data)
- Append new data and train, if not anomalous
- 80-90% validation
- Train on “Healthy“ data 12 hours before to test new incoming data

Machine Learning Verdict			
Linear Regressions	SVM	IsolationForest	Autoencoder
Normal	Normal	Normal	Normal
Normal	Normal	Normal	Normal
Normal	ANOMALY	Normal	Normal
Normal	Normal	Normal	Normal
Normal	Normal	Normal	Normal
Normal	ANOMALY	Normal	Normal
FanRMSE: +/- 1759.33	Normal	ANOMALY	ANOMALY
nvme Temp RMSE: +/- 14755.6	Normal	ANOMALY	ANOMALY
Normal	Normal	ANOMALY	Normal
Normal	Normal	ANOMALY	ANOMALY
Normal	Normal	ANOMALY	Normal
FanRMSE: +/- 3552.727	Normal	ANOMALY	Normal
Normal	Normal	ANOMALY	Normal
Normal	Normal	ANOMALY	Normal
Normal	Normal	ANOMALY	Normal
FanRMSE: +/- 1260793.473	Normal	ANOMALY	Normal

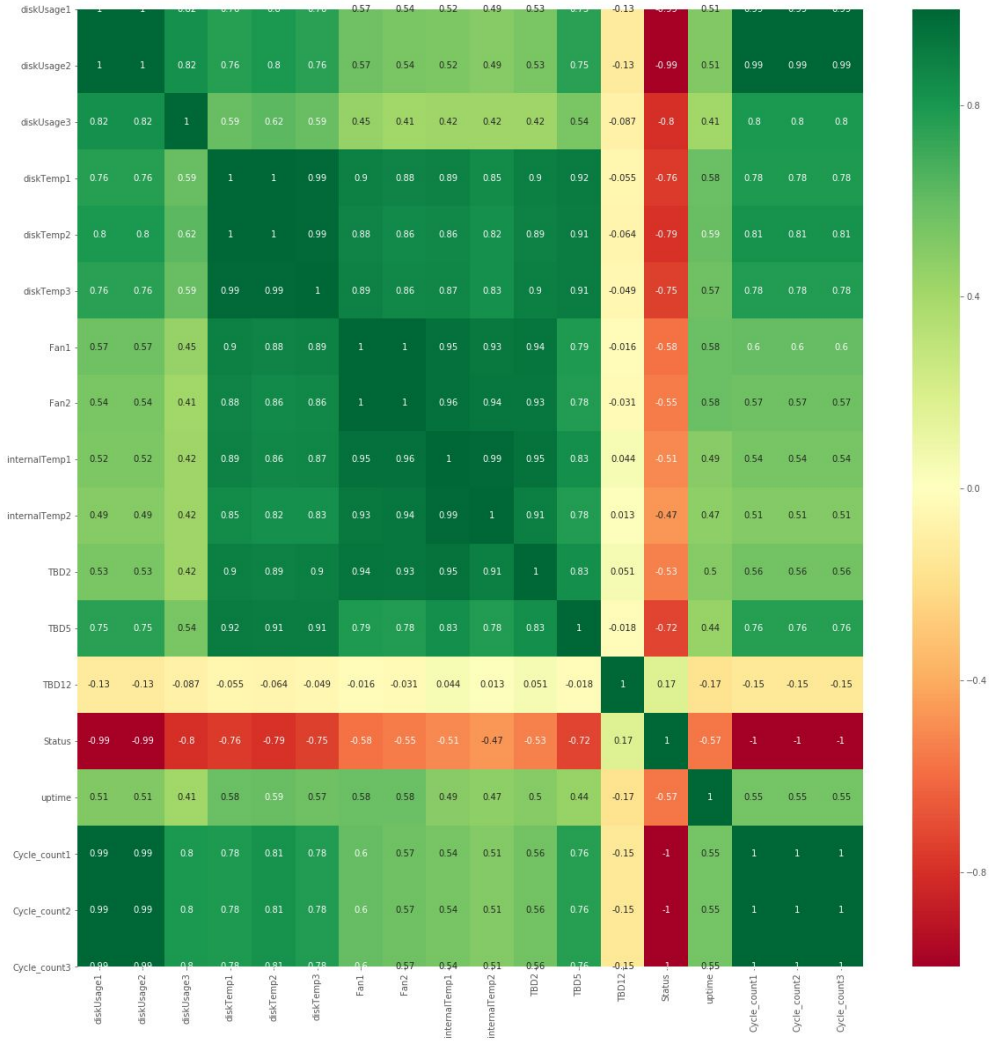
Data Preprocessing

Scalers

- Re-scale data (eg: normalized, min-max scaling)
- Some models train better on different methods of scaled data, giving better results

Principal Component Analysis (PCA)

- Reduce dimensionality of training set by analysing relations between variables
- Determining which variables skew the data the most
- Can make ML learning more efficient and run faster
- Use Linear regressions to find Variance Inflation Factor (VIF) and remove components of VIF>5

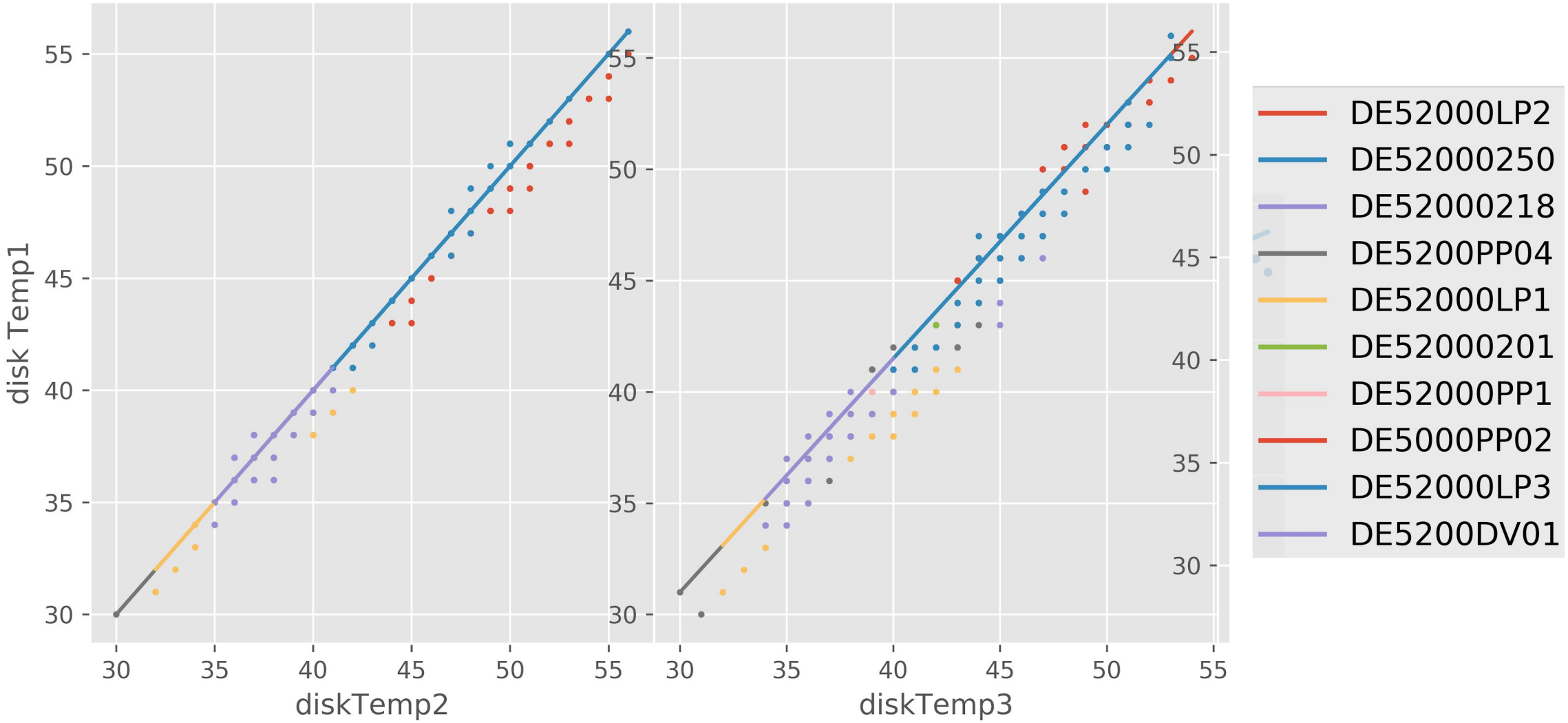


IU Temp

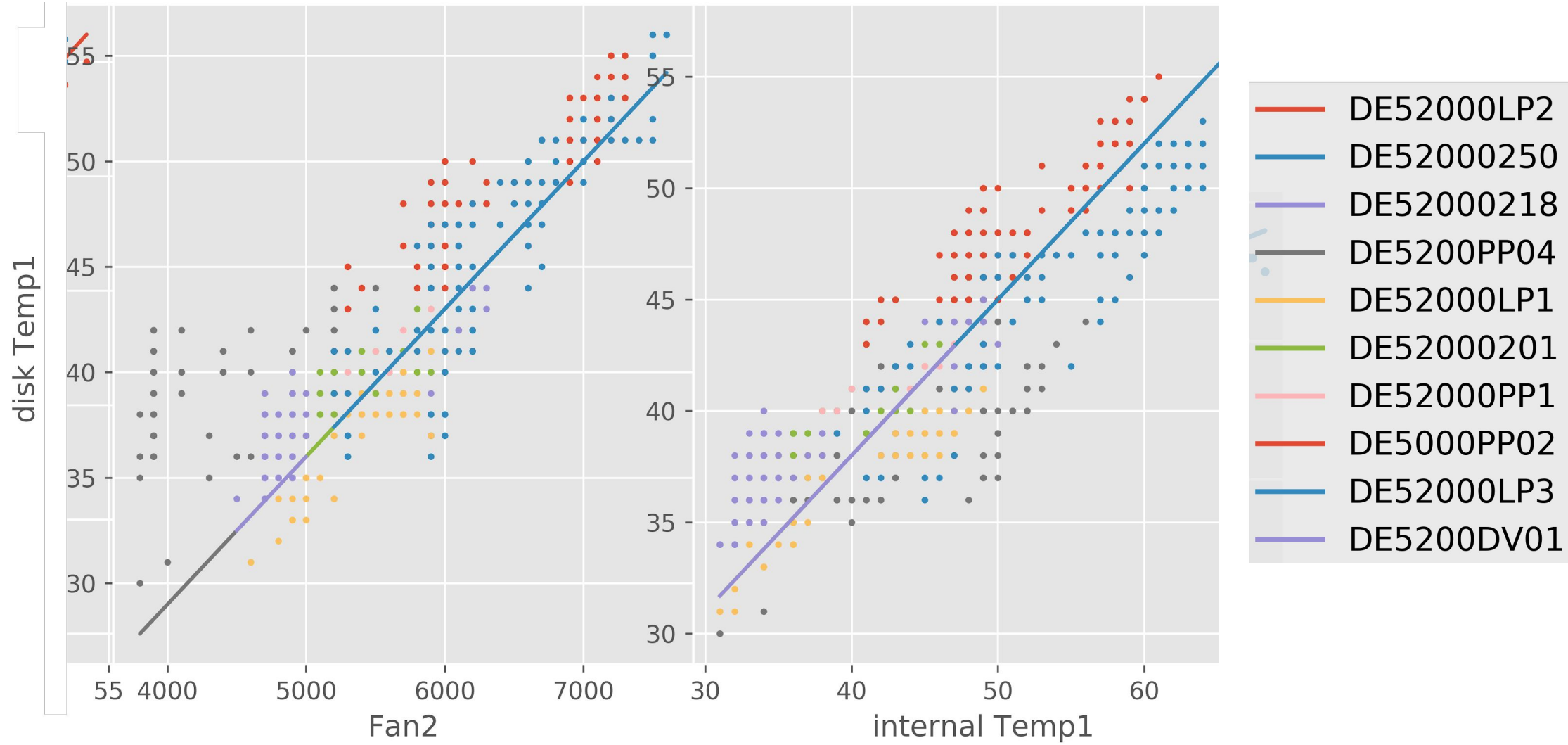
PU Temp

IU Status

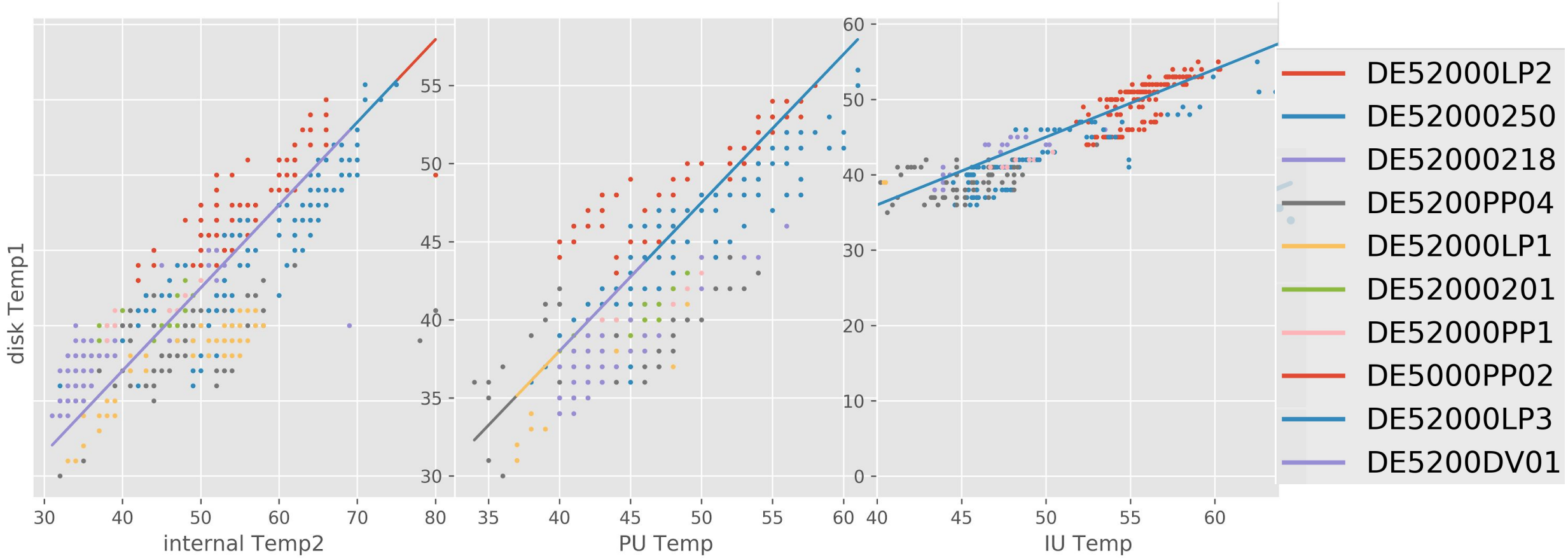
Data Preprocessing: Linear Regression Analysis



Data Preprocessing: Linear Regression Analysis



Data Preprocessing: Linear Regression Analysis



Telemetry Dashboard #1: Machine Learning Columns

Linear Regressions:

- Sklearn Linear Regression model
- No scaling or PCA
- Checks predictions for
  - InternalTemp1
  - Fan1
  - PU Temp

- Flags if RMSE > 10% of actual value, per three variables

Machine Learning Verdict			
Linear Regressions	SVM	IsolationForest	Autoencoder
Normal	Normal	Normal	Normal
Normal	Normal	Normal	Normal
Normal	ANOMALY	Normal	Normal
Normal	Normal	Normal	Normal
Normal	Normal	Normal	Normal
Normal	ANOMALY	Normal	Normal
FanRMSE: +/- 1759.33	Normal	ANOMALY	ANOMALY
nvme Temp RMSE: +/- 14755.6	Normal	ANOMALY	ANOMALY
Normal	Normal	ANOMALY	Normal
Normal	Normal	ANOMALY	ANOMALY
Normal	Normal	ANOMALY	Normal
FanRMSE: +/- 3552.727	Normal	ANOMALY	Normal
Normal	Normal	ANOMALY	Normal
Normal	Normal	ANOMALY	Normal
Normal	Normal	ANOMALY	Normal
FanRMSE: +/- 1260793.473	Normal	ANOMALY	Normal

Support Vector Machine (SVM)

- SKLearn ML model that takes labeled training data
- Outputs an “optimal hyperplane”, categorizes new examples

1 Class SVM

- Unsupervised outlier detection
- Estimates the support of a high-dimensional distribution
- Trains on “Healthy” class of data
- Tests new input data to determine if it fits within criteria of trained data
- Train on this new data if it is determined to be healthy, updates model

Telemetry Dashboard #1: Machine Learning Columns

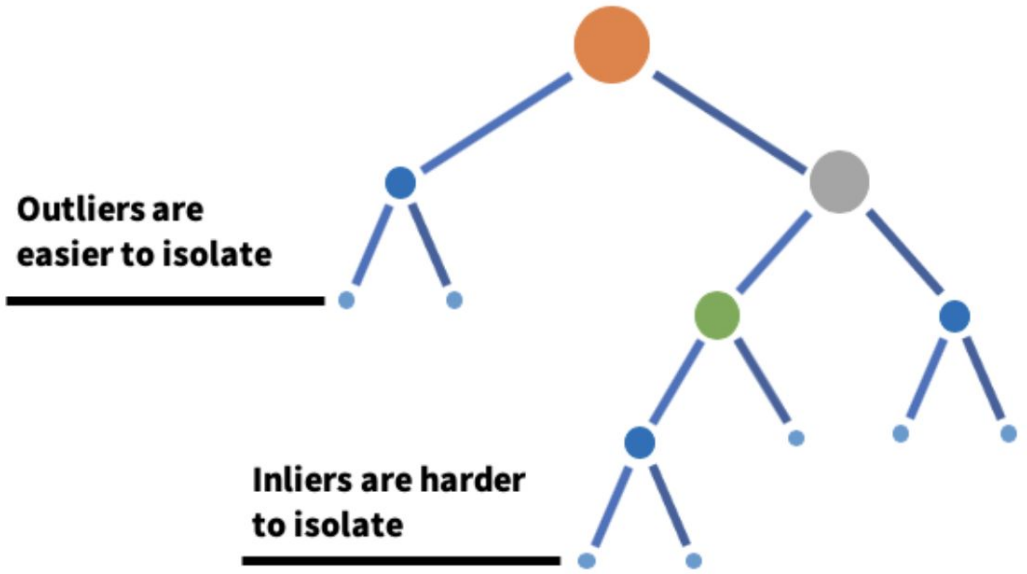
Machine Learning Verdict			
Linear Regressions	SVM	IsolationForest	Autoencoder
Normal	Normal	Normal	Normal
Normal	Normal	Normal	Normal
Normal	ANOMALY	Normal	Normal
Normal	Normal	Normal	Normal
Normal	Normal	Normal	Normal
Normal	ANOMALY	Normal	Normal
FanRMSE: +/- 1759.33	Normal	ANOMALY	ANOMALY
nvme Temp RMSE: +/- 14755.6	Normal	ANOMALY	ANOMALY
Normal	Normal	ANOMALY	Normal
Normal	Normal	ANOMALY	ANOMALY
Normal	Normal	ANOMALY	Normal
FanRMSE: +/- 3552.727	Normal	ANOMALY	Normal
Normal	Normal	ANOMALY	Normal
Normal	Normal	ANOMALY	Normal
Normal	Normal	ANOMALY	Normal
FanRMSE: +/- 1260793.473	Normal	ANOMALY	Normal

1 Class SVM:

- min_max_scaler
- PCA for 2 components
- Nu parameter: 0.1
- gamma='auto'

- Anomaly if SVM returns -1

Isolation Forest



- SKLearn ML anomaly detection model
- Isolates observations by randomly selecting a feature
  - then randomly selecting a split value between max and min values of that feature
- “Recursive partitioning” represented by a tree structure:
  - ➢ Number of splittings required to isolate a sample == **path length** from the root node to the terminating node
  - ➢ **Path length**, averaged over a forest of random trees == measure of normality and decision function
- Random partitioning produces noticeably shorter paths for anomalies
  - ➢ If forest of random trees all produce shorter path lengths for samples, are likely to be anomalies

Telemetry Dashboard #1: Machine Learning Columns

Machine Learning Verdict			
Linear Regressions	SVM	IsolationForest	Autoencoder
Normal	Normal	Normal	Normal
Normal	Normal	Normal	Normal
Normal	ANOMALY	Normal	Normal
Normal	Normal	Normal	Normal
Normal	Normal	Normal	Normal
Normal	ANOMALY	Normal	Normal
FanRMSE: +/- 1759.33	Normal	ANOMALY	ANOMALY
nvme Temp RMSE: +/- 14755.6	Normal	ANOMALY	ANOMALY
Normal	Normal	ANOMALY	Normal
Normal	Normal	ANOMALY	ANOMALY
Normal	Normal	ANOMALY	Normal
FanRMSE: +/- 3552.727	Normal	ANOMALY	Normal
Normal	Normal	ANOMALY	Normal
Normal	Normal	ANOMALY	Normal
Normal	Normal	ANOMALY	Normal
FanRMSE: +/- 1260793.473	Normal	ANOMALY	Normal

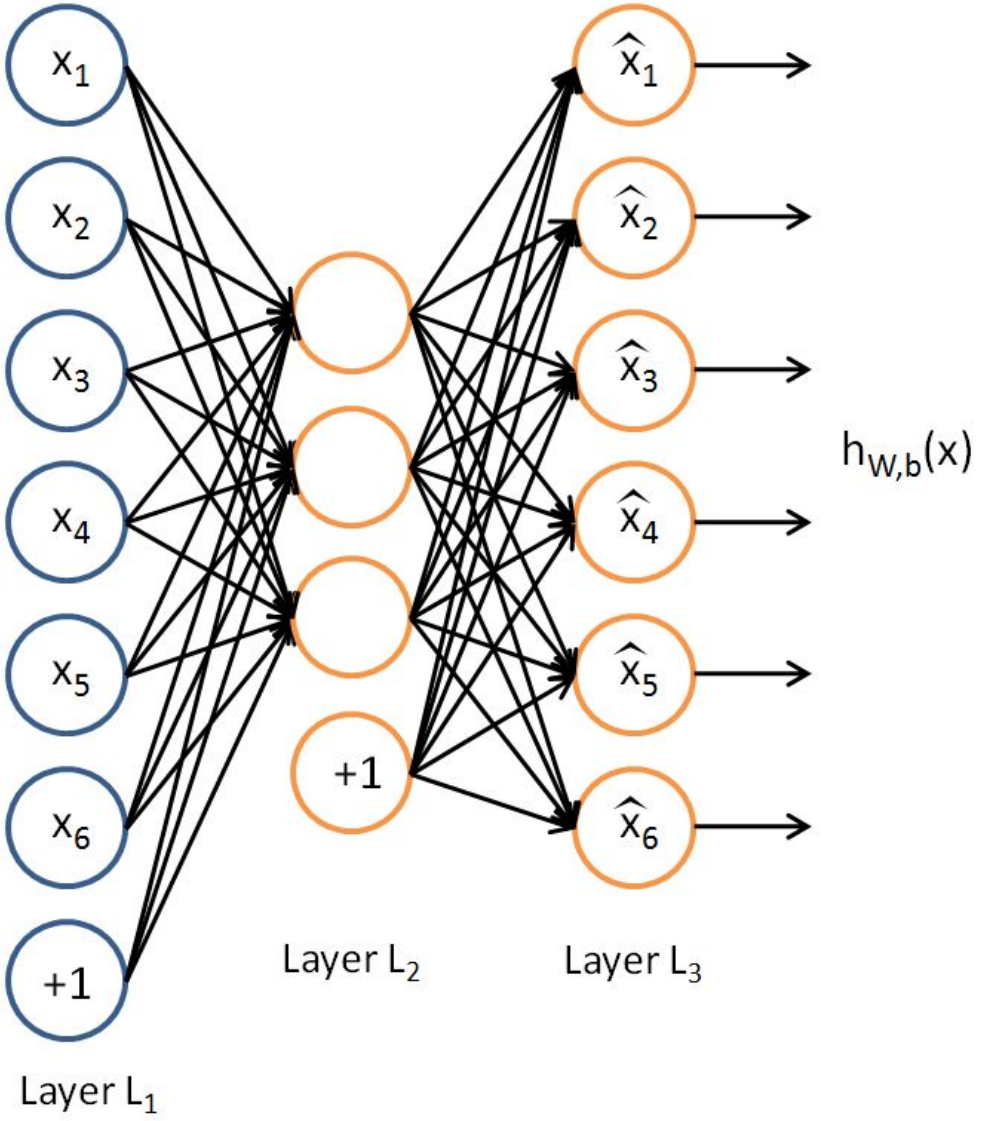
IsolationForest:

- min_max_scaler
- PCA for 2 components

- Anomaly if returns -1

Autoencoders

- Artificial neural network
- Unsupervised ML model that learns from input data
- Given unlabeled training examples
- Applies backpropagation, sets target values to be equal to the inputs
- Gain insight on structure of the data by applying constraints to the network: limit number of hidden layers



Autoencoders

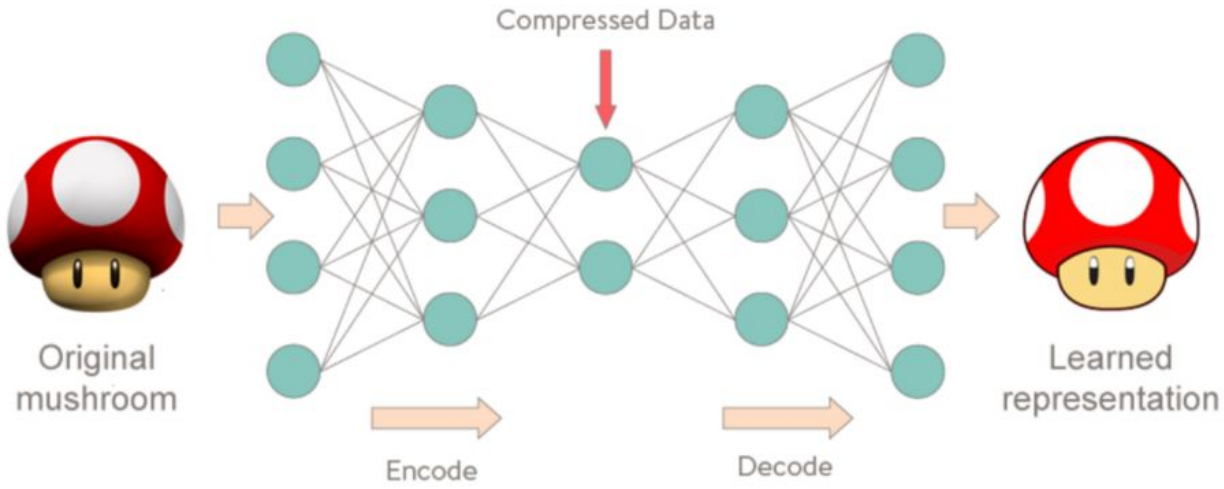
- For small number of hidden units, network forced to learn a ”compressed” representation of the input, must try to ”‘reconstruct”’ the input

- For random input, compression task would be very difficult.

- But if there is structure in data, eg: correlated input features, then this algorithm will be able to discover some of those correlations.

- Simple autoencoder learns a low-dimensional representation similar to PCAs.

- “Overkill” for simple data, can lead to over-fitting



Telemetry Dashboard #1: Machine Learning Columns

Machine Learning Verdict			
Linear Regressions	SVM	IsolationForest	Autoencoder
Normal	Normal	Normal	Normal
Normal	Normal	Normal	Normal
Normal	ANOMALY	Normal	Normal
Normal	Normal	Normal	Normal
Normal	Normal	Normal	Normal
Normal	ANOMALY	Normal	Normal
FanRMSE: +/- 1759.33	Normal	ANOMALY	ANOMALY
nvme Temp RMSE: +/- 14755.6	Normal	ANOMALY	ANOMALY
Normal	Normal	ANOMALY	Normal
Normal	Normal	ANOMALY	ANOMALY
Normal	Normal	ANOMALY	Normal
FanRMSE: +/- 3552.727	Normal	ANOMALY	Normal
Normal	Normal	ANOMALY	Normal
Normal	Normal	ANOMALY	Normal
Normal	Normal	ANOMALY	Normal
FanRMSE: +/- 1260793.473	Normal	ANOMALY	Normal

Autoencoder:

- min_max_scaler
- No PCA
- 9 initial layers, 1 per feature

```
min_max_scaler.fit(train_X)
# Apply transform to both the training set and the test set.


train_X = min_max_scaler.transform(train_X)
test_X = min_max_scaler.transform(test_X)

# No of Neurons in each Layer [9,6,3,2,3,6,9]
input_dim = train_X.shape[1]
encoding_dim = 6
input_layer = Input(shape=(input_dim, ))
encoder = Dense(encoding_dim, activation="tanh",activity_regularizer=regularizers.l1(10e-5))(input_layer)
encoder = Dense(int(encoding_dim / 2), activation="tanh")(encoder)
encoder = Dense(int(2), activation="tanh")(encoder)
decoder = Dense(int(encoding_dim/ 2), activation='tanh')(encoder)
decoder = Dense(int(encoding_dim), activation='tanh')(decoder)
decoder = Dense(input_dim, activation='tanh')(decoder)
autoencoder = Model(inputs=input_layer, outputs=decoder)
autoencoder.compile(optimizer='adam', loss='mse' )
history = autoencoder.fit(train_X, train_X,epochs=nb_epoch,batch_size=batch_size,shuffle=True,validation_split=0.1,verbose=0)
df_results=output
df_results['AAMSE']=np.mean(np.power(test_X - autoencoder.predict(test_X), 2), axis=1)
```

- Anomaly if MSE>0.1

Telemetry Dashboard #2:

Green: healthy point, predicted by autoencoder

Orange: Error point based on TBD12 readout

Disk 1,2 usages

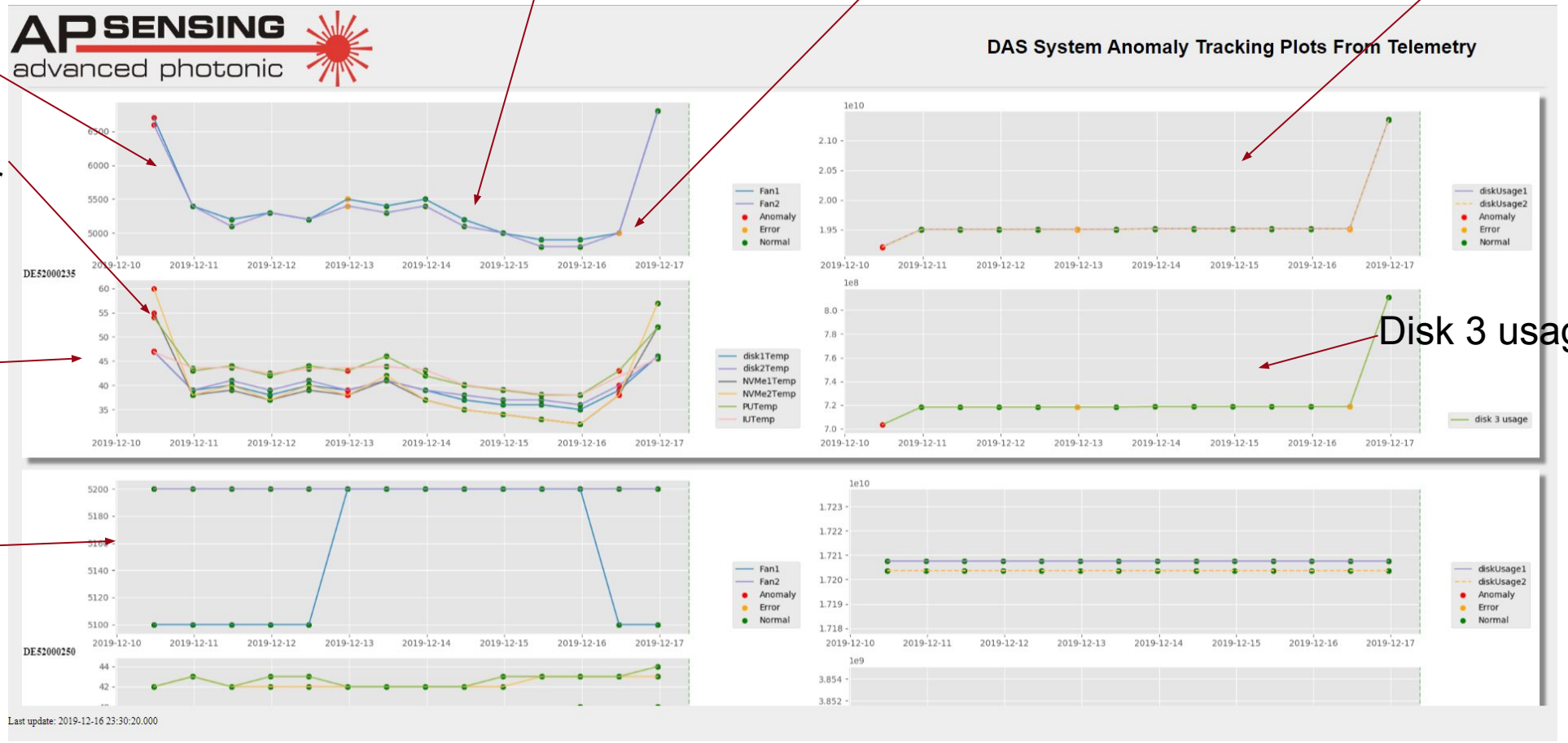
Fan speeds

Red: Anomaly point based on Autoencoder model

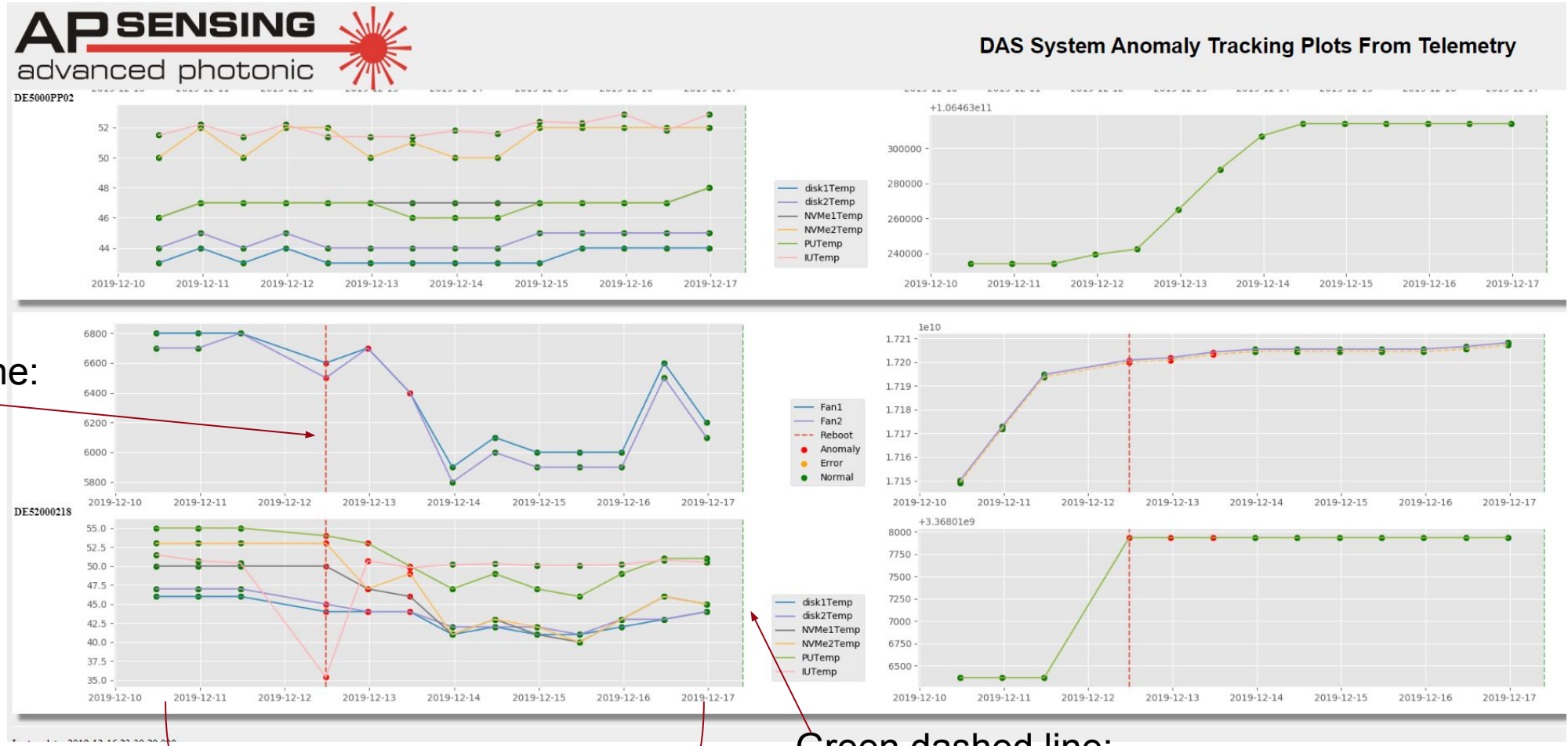
Temperatures

Disk 3 usag

Plots per ID active in last 6 days



Telemetry Dashboard #2:



Red dashed line:
Hard Reboot

Green dashed line:
today

Last 7 days

Project II: Machine Learning Library Upload and Dashboard

- DAS systems alarm based on event types: Trains, Cars, Cable thefts
- Different ML libraries built to classify events
- Visualizations of events and model performance from updating database
  - Automated and uploaded to dashboard

22

Machine Learning Library: Upload Data Script

- ‘Upload_Data.py’, gives hdf5 file location
- Takes features from Feature_Calculation file, feature_sizes
- Class from Folder names
- Sitename based on “Metadata/Country”
- Generates hash id, based on file name and row number, allows for overwriting on second upload
- Row is just index
- If file unspecified, will run on all ML library files

```
import numpy
import pandas as pd
from elasticsearch import Elasticsearch
from elasticsearch.helpers import bulk
from datetime import date
import glob
import os
import sys
import hashlib
from codecs import encode
es = Elasticsearch(hosts="10.0.12.4", timeout=10000)
sys.path.insert(1, '../ML_features/')
from Feature_Calculation import feature_sizes

def KibanaHDF5Upload(hdf5_fileloc):
    """
    Function to upload hdf5 file data to Elasticsearch index: ml_library_data.
    Give the path location of the file, else it will run on all files in the ML_Library folder on raid0.
    eg:
        KibanaHDF5Upload('/mnt/raid0/ML_Library/Car/Prci_Car_1_7/Prci_Car_1_7.hdf5')
    """
    df=pd.DataFrame()
    with h5py.File(hdf5_fileloc,"r",libver="latest") as h5_file:
        for key in feature_sizes:
            if feature_sizes[key]>1:
                feat_list=[]
                for i in range(len(h5_file["features/"+key][()])):
                    feat=h5_file["features/"+key][i]
                    feat_list.append(feat)
                featdf=pd.DataFrame(feat_list)
                for i in range(len(featdf.columns)):
                    featdf=featdf.rename(columns={i : str(key)+"_"+str(i)})
                df=pd.concat([df,featdf],axis=1)
            else:
                featdf=pd.DataFrame(h5_file["features/"+key][()]).rename(columns={0 : key})
                df=pd.concat([df,featdf],axis=1)
        ids=[]
        filename=os.path.basename(hdf5_fileloc)
        class_=os.path.split(os.path.split(os.path.dirname(hdf5_fileloc))[0])[-1]
        try: Sitename=str(h5_file["Metadata/Country"][()][0])
        except KeyError: Sitename="unknown"
        name=[]
        row=[]
        classtype=[]
        site=[]
        UploadDate=[]
        for i in range(len(df)):
            ids.append(str(hashlib.sha224(encode(filename)).hexdigest())[:11]+"_"+str("{:08d}".format(i)))
            name.append(filename)
            row.append(i)
            classtype.append(class_)
            site.append(Sitename)
            UploadDate.append(date.today())
        ids=pd.DataFrame(ids).rename(columns={0 : "_id"})
        name=pd.DataFrame(name).rename(columns={0 : "Filename"})
        UploadDate=pd.DataFrame(UploadDate).rename(columns={0 : "UploadDate"})
        row=pd.DataFrame(row).rename(columns={0 : "Row"})
        classtype=pd.DataFrame(classtype).rename(columns={0 : "class"})
        site=pd.DataFrame(site).rename(columns={0 : "Site"})

        df=pd.concat([df,name,UploadDate, row, ids,classtype, site],axis=1)
    dicts = df.to_dict(orient='records')
    bulk(es,  dicts, index="ml_library_data", doc_type="_doc")

if __name__ == "__main__":
    from tqdm.autonotebook import trange
    ml_file_list = glob.glob('/mnt/raid0/ML_Library/*/*/*.hdf5')
    t=trange(len(ml_file_list))
    for file in t:
        fname = ml_file_list[file]
        t.set_description("Uploading "+str(fname))
        KibanaHDF5Upload(fname)
```

Machine Learning Library: Dashboard

- 'plots.py' running on **Venus**, pulls Elasticsearch data makes plots for Dashboard, curl plots to **Pluto**
- 'script.py' running on **Pluto** (/usr/lib/cgi-bin/ml_dashboard/, ith plots read from /var/www/html/ml_plots simlink to home/ml_plots)

Machine Learning Library: plots.py

Filename list from data

```
res = es.search(index="ml_library_data", body=
#Put request in here#
{
  "aggs": {
    "3": {
      "terms": {
        "field": "Filename.keyword",
        "order": {
          "_count": "desc"
        },
        "size": 200
      }
    }
  }
}
fname=[]
for i in res["aggregations"]["3"]["buckets"]:
    fname.append(i["key"])
res = es.search(index="ml_library_data", body=
{
  "aggs": {
    "3": {
      "terms": {
        "field": "Class.keyword",
        "order": {
          "_count": "desc"
        },
        "size": 200
      }
    }
  }
}
cname=[]
for i in res["aggregations"]["3"]["buckets"]:
    cname.append(i["key"])
```

```
['IMO_9296195.hdf5',
 'IMO_9131797_validation.hdf5',
 'IMO_9242730.hdf5',
 'IMO_9206671.hdf5',
 'IMO_9229879.hdf5',
 'IMO_9010163.hdf5',
 'IMO_9336268.hdf5',
 'IMO_9217230.hdf5',
 'IMO_9760512.hdf5',
 'IMO_9215505.hdf5',
 'DB_cableTheft_3.hdf5',
 'DB_cableTheft_1.hdf5',
 'DB_cableTheft_2.hdf5',
 'DB_cableTheft_4.hdf5',
 'DB_cableTheft_5.hdf5',
 'DB_cableTheft_6.hdf5',
 'IMO_9650482.hdf5',
 'IMO_9319454.hdf5',
 'IMO_9164524.hdf5',
 'IMO_9491501.hdf5',
 'DB_Berlin_cableTheft10.hdf5',
 'DB_Berlin_cableTheft7.hdf5',
 'DB_cableTheft_blindTrial_02.hdf5',
 'IMO_9568249.hdf5',
 'DB_cableTheft_blindTrial_01.hdf5',
 'DB_Berlin_cableTheft9.hdf5',
 'IMO_9666431.hdf5',
 'DB_Berlin_cableTheft8.hdf5',
 'IMO_5302063.hdf5']
```

Class name list from data

```
['Train',
 'Ship',
 'Ship_Background',
 'Track_Distance',
 'Car',
 'Digging_Mechanical',
 'Prysmian_Digging',
 'Prysmian_Train',
 'CableTheft',
 'Walking',
 'Boring',
 'Orsted',
 'RTE_Version7_Cars',
 'Drilling',
 'Digging_Manual',
 'Aggriculture_Machine',
 'Dropping']
```

```
dictc={}
for name in cname:
    match_phrase={"Class.keyword": str(name)}
    match_phrase={"match_phrase":match_phrase}
    bool={"should":[match_phrase]}
    filter={"bool":bool}
    filter=[filter]
    nbool={"must":[],"filter":filter}
    nbool={"bool":nbool}
    nfilt={str(name):nbool}
    dictc[name]=nbool
##################################################
```

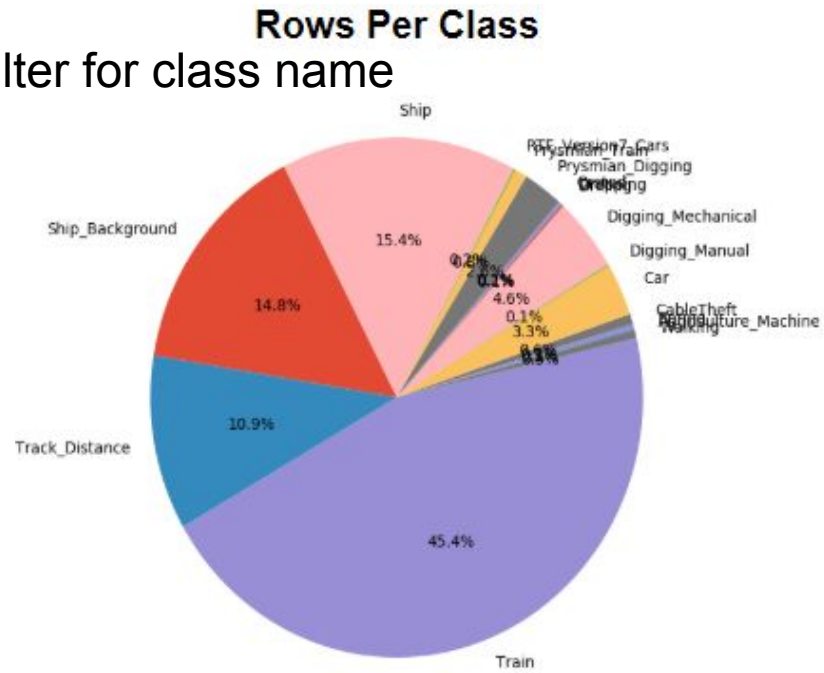
Filter for class name

```
res = es.search(index="ml_library_data", body=
#Put request in here#
{
"aggs": {
    "2": {
      "filters": {
#####################################
        "filters": dictc
#####################################
                }
      }
    }
  }
            )

Y1=[]
Y2=[]

Class=cname
#print(Class)
colors=[]
explode=[]
j=0
for i in cname:
    #X=res["aggregations"]["3"]["buckets"][i]
    y1=res["aggregations"]["2"]["buckets"][i]['doc_count']   #FBE1
    #y2=res["aggregations"]["3"]["buckets"][i]["4"]['value']   #FBE2
    Y1.append(float(y1 or 0))
    #Y2.append(float(y2 or 0))
    j+=1
    colors.append("C"+str(j))
    explode.append(y1*0.000001)
    #print(y1)
plt.figure(figsize=(9.5,6))
plt.pie(Y1, colors=colors, autopct='%1.1f%%', startangle=15,labels=Class)

#plt.legend(labels=Class, loc=(0.9,0.2))
plt.tight_layout()
plt.savefig('ClassPie.png')
os.system('curl --insecure --user pi:APSdas123 -T ClassPie.png sftp://10.0.12.10/~/ml_plots/ClassPie.png')
plt.clf()
################################
```



Machine Learning Library: plots.py

```
##################################################
res = es.search(index="ml_library_data", body=
#Put request in here#
{
  "aggs": {
    "3": {
      "terms": {
        "field": "Filename.keyword",
        "order": {
          "_count": "desc"
        },
        "size": 200
      }
    }
  }
}
)
Fname=[]
for i in res["aggregations"]["3"]["buckets"]:
    Fname.append(i["key"])
res = es.search(index="ml_library_data", body=
{
  "aggs": {
    "3": {
      "terms": {
        "field": "Class.keyword",
        "order": {
          "_count": "desc"
        },
        "size": 200
      }
    }
  }
}
)
cname=[]
for i in res["aggregations"]["3"]["buckets"]:
    cname.append(i["key"])
##################################################
```

```
['IMO_9296195.hdf5',
 'IMO_9131797_validation.hdf5',
 'IMO_9242730.hdf5',
 'IMO_9206671.hdf5',
 'IMO_9229879.hdf5',
 'IMO_9010163.hdf5',
 'IMO_9336268.hdf5',
 'IMO_9217230.hdf5',
 'IMO_9760512.hdf5',
 'IMO_9215505.hdf5',
 'DB_CableTheft_3.hdf5',
 'DB_CableTheft_1.hdf5',
 'DB_CableTheft_2.hdf5',
 'DB_CableTheft_4.hdf5',
 'DB_CableTheft_5.hdf5',
 'DB_CableTheft_6.hdf5',
 'IMO_9650482.hdf5',
 'IMO_9319454.hdf5',
 'IMO_9164524.hdf5',
 'IMO_9491501.hdf5',
 'DB_Berlin_cableTheft10.hdf5',
 'DB_Berlin_cableTheft7.hdf5',
 'DB_cableTheft_blindTrial_02.hdf5',
 'IMO_9568249.hdf5',
 'DB_cableTheft_blindTrial_01.hdf5',
 'DB_Berlin_cableTheft9.hdf5',
 'IMO_9666431.hdf5',
 'DB_Berlin_cableTheft8.hdf5',
 'IMO_5302063.hdf5']
```

```
['Train',
 'Ship',
 'Ship_Background',
 'Track_Distance',
 'Car',
 'Digging_Mechanical',
 'Prysmian_Digging',
 'Prysmian_Train',
 'CableTheft',
 'Walking',
 'Boring',
 'Orsted',
 'RTE_Version7_Cars',
 'Drilling',
 'Digging_Manual',
 'Aggriculture_Machine',
 'Dropping']
```

Class name list from data

```
dictc={}
for name in cname:
    match_phrase={"Class.keyword": str(name)}
    match_phrase={"match_phrase":match_phrase}
    bool={"should":[match_phrase]}
    filter={"bool":bool}
    filter=[filter]
    nbool={"must":[],"filter":filter}
    nbool={"bool":nbool}
    nfilt={str(name):nbool}
    dictc[name]=nbool
##################################################
```

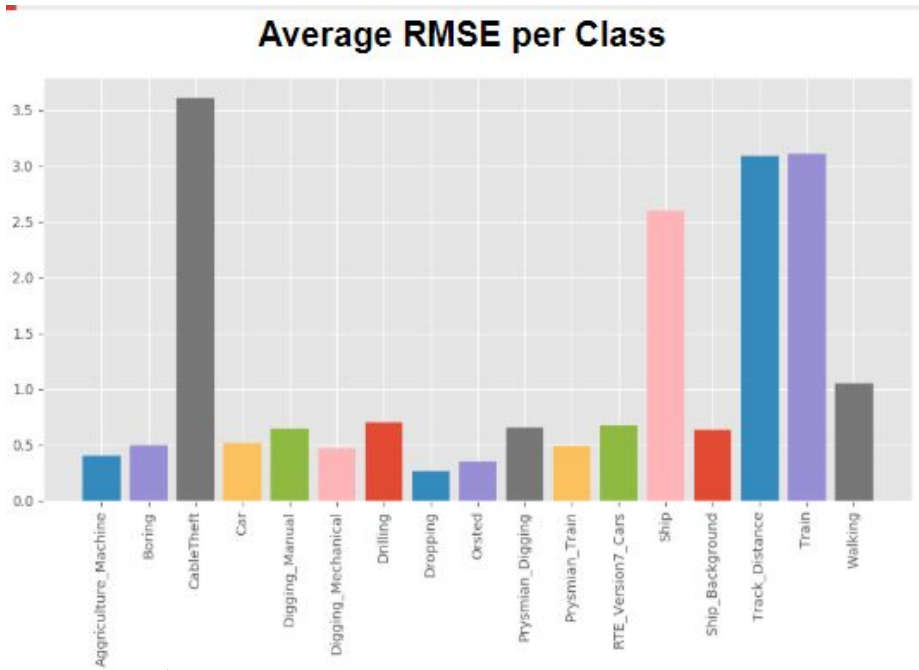
```
res = es.search(index="ml_library_data", body=
#Put request in here#
{
"aggs": {
    "3": {
      "filters": {
        ##Class Filters##
##################################
"filters": dictc
##################################
      },
      "aggs": {
        "2": {
          "avg": {
            "field": "rms"
          }
        }
      }
    }
  }
}
            )

Y1=[]
Y2=[]
colors=[]
j=0
Class=cname
for i in cname:
    y1=res["aggregations"]["3"]["buckets"][i]["2"]['value']   #FBE1
    Y1.append(float(y1 or 0))
    j+=1
    colors.append("C"+str(j))

y_pos=np.arange(len(Class))
plt.bar(y_pos, Y1,color=colors)
plt.xticks(y_pos, Class,rotation='vertical')

plt.tight_layout()
plt.savefig('RMSE_Histo.png')
os.system('curl --insecure --user pi:APSdas123 -T RMSE_Histo.png sftp://10.0.12.10/~/ml_plots/RMSE_Histo.png')
plt.clf()
#####################################
```

"rms"



Machine Learning Library: plots.py

```
######################################################
res = es.search(index="ml_library_data", body=
#Put request in here#
{
  "aggs": {
    "3": {
      "terms": {
        "field": "Filename.keyword",
        "order": {
          "_count": "desc"
        },
        "size": 200
      }
    }
  }
}
)
Fname=[]
for i in res["aggregations"]["3"]["buckets"]:
    Fname.append(i["key"])
res = es.search(index="ml_library_data", body=
{
  "aggs": {
    "3": {
      "terms": {
        "field": "Class.keyword",
        "order": {
          "_count": "desc"
        },
        "size": 200
      }
    }
  }
}
)
cname=[]
for i in res["aggregations"]["3"]["buckets"]:
    cname.append(i["key"])
######################################################
```

Class name list from data

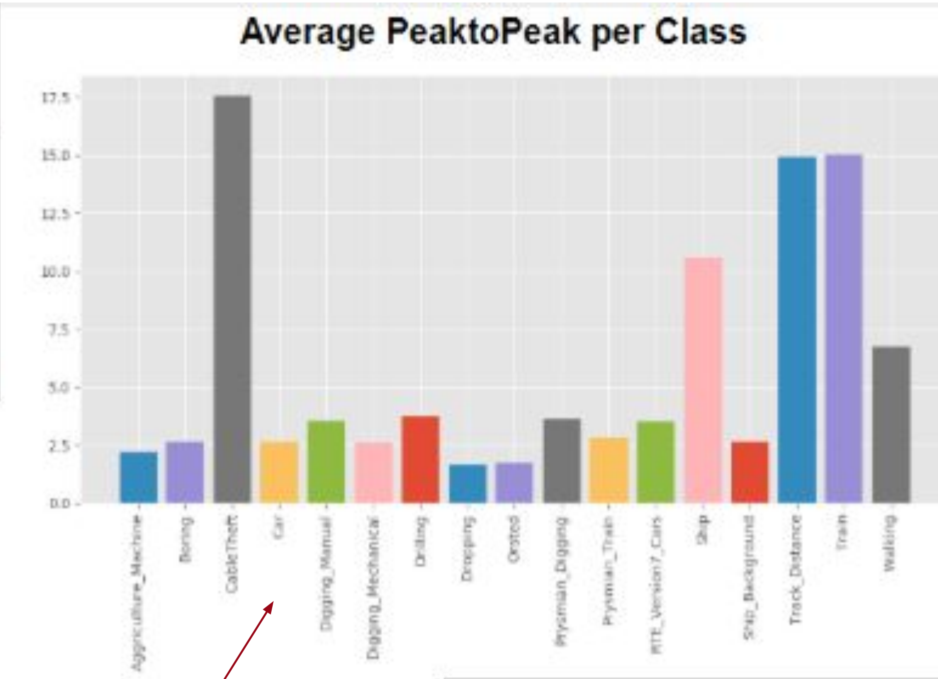
```
['IMO_9296105.hdf5',
 'IMO_9131797_validation.hdf5',
 'IMO_9242730.hdf5',
 'IMO_9206671.hdf5',
 'IMO_9229879.hdf5',
 'IMO_9010163.hdf5',
 'IMO_9336268.hdf5',
 'IMO_9217230.hdf5',
 'IMO_9760512.hdf5',
 'IMO_9215505.hdf5',
 'DB_cableTheft_3.hdf5',
 'DB_cableTheft_1.hdf5',
 'DB_cableTheft_2.hdf5',
 'DB_cableTheft_4.hdf5',
 'DB_cableTheft_5.hdf5',
 'DB_cableTheft_6.hdf5',
 'IMO_9650482.hdf5',
 'IMO_9319454.hdf5',
 'IMO_9164524.hdf5',
 'IMO_9491581.hdf5',
 'DB_Berlin_cableTheft10.hdf5',
 'DB_Berlin_cableTheft7.hdf5',
 'DB_cableTheft_blindTrial_02.hdf5',
 'IMO_9568249.hdf5',
 'DB_cableTheft_blindTrial_01.hdf5',
 'DB_Berlin_cableTheft9.hdf5',
 'IMO_9666431.hdf5',
 'DB_Berlin_cableTheft8.hdf5',
 'IMO_5302063.hdf5']
```

```
['Train',
 'Ship',
 'Ship_Background',
 'Track_Distance',
 'Car',
 'Digging_Mechanical',
 'Prysmian_Digging',
 'Prysmian_Train',
 'CableTheft',
 'Walking',
 'Boring',
 'Orsted',
 'RTE_Version7_Cars',
 'Drilling',
 'Digging_Manual',
 'Aggriculture_Machine',
 'Dropping']
```

```
dictc={}
for name in cname:
    match_phrase={"Class.keyword": str(name)}
    match_phrase={"match_phrase":match_phrase}
    bool={"should":[match_phrase]}
    filter={"bool":bool}
    filter=[filter]
    nbool={"must":[],"filter":filter}
    nbool={"bool":nbool}
    nfilt={str(name):nbool}
    dictc[name]=nbool
######################################################
```

```
######################################################
res = es.search(index="ml_library_data", body=
#Put request in here#
{
"aggs": {
    "3": {
      "filters": {
######################################################
          "filters": dictc
######################################################
      },
      "aggs": {
        "2": {
          "avg": {
            "field": "peak_to_peak"
          }
        }
      }
    }
  }
}
              )
Y1=[]
Y2=[]
colors=[]
j=0
Class=cname
for i in cname:
    #x=res["aggregations"]["3"]["buckets"][i]
    y1=res["aggregations"]["3"]["buckets"][i]["2"]['value']    #FBE1
    #y2=res["aggregations"]["3"]["buckets"][i]["4"]['value']    #FBE2
    Y1.append(float(y1 or 0))
    #Y2.append(float(y2 or 0))
    j+=1
    colors.append("C"+str(j))
#colors=["#0B0442","#E24433","#FF8500","#F8C15F","#988ED5","#348ABD"]
y_pos=np.arange(len(Class))
plt.bar(y_pos, Y1,color=colors)
plt.xticks(y_pos, Class,rotation='vertical')
plt.tight_layout()
plt.savefig('PeaktoPeak_Histo.png')
os.system('curl --insecure --user pi:APSdas123 -T PeaktoPeak_Histo.png sftp://10.0.12.10/~/ml_plots/PeaktoPeak_Histo.png')
plt.clf()
######################################################
```

"peak_to_peak"



Machine Learning Library: plots.py

```
dict={}
for i in range(20):
    field={"field": "MFCC_"+str(i)}
    ext={"extended_stats":field}
    dict[str(i+1)]=ext
agg={"aggs":dict}

res = es.search(index="ml_library_data", body=agg)
X=[]
Yavg=[]
YstdUp=[]
YstdDown=[]
Ymax=[]
Ymin=[]
x=res["aggregations"].keys()
for i in (x):
    X.append(int(i))
    Yavg.append(res["aggregations"][i]["avg"])
    YstdUp.append(res["aggregations"][i]["avg"]+res["aggregations"][i]["std_deviation"])
    YstdDown.append(res["aggregations"][i]["avg"]-res["aggregations"][i]["std_deviation"])
    Ymax.append(res["aggregations"][i]["max"])
    Ymin.append(res["aggregations"][i]["min"])
Avg=pd.DataFrame({"Average" : Yavg},X).sort_index()
StdeUp=pd.DataFrame({"Upper Standard Deviation" :YstdUp},X).sort_index()
StdeDown=pd.DataFrame({"Lower Standard Deviation" :YstdDown},X).sort_index()
MFCCMax=pd.DataFrame({"Max" :Ymax},X).sort_index()
MFCCMin=pd.DataFrame({"Min" :Ymin},X).sort_index()


plt.figure(figsize=(9.5,6))
plt.plot(Avg, color="#777777", label='Average')
plt.plot(StdeUp, color="#FBC15E", label='Standard Deviation')
plt.plot(StdeDown, color="#FBC15E")
plt.plot(MFCCMax, color='#E24A33', label='Max')
plt.plot(MFCCMin, color="#FFB5B8", label='Min')

plt.fill_between(Avg.index,MFCCMax["Max"],StdeUp["Upper Standard Deviation"], color='#E24A33')
plt.fill_between(Avg.index,Avg["Average"],StdeUp["Upper Standard Deviation"], color='#FBC15E')
plt.fill_between(Avg.index,Avg["Average"],StdeDown["Lower Standard Deviation"], color='#FBC15E')
plt.fill_between(Avg.index,StdeDown["Lower Standard Deviation"],MFCCMin["Min"], color='#FFB5B8')
plt.xlim(1,20)
plt.xlabel("MFCC Element")
plt.ylabel("MFCC")
plt.legend()
plt.tight_layout()
plt.savefig('MFCC.png')
os.system('curl --insecure --user pi:APSdas123 -T MFCC.png sftp://10.0.12.10/~/ml_plots/MFCC.png')
plt.clf()
```

All “MFCC” values

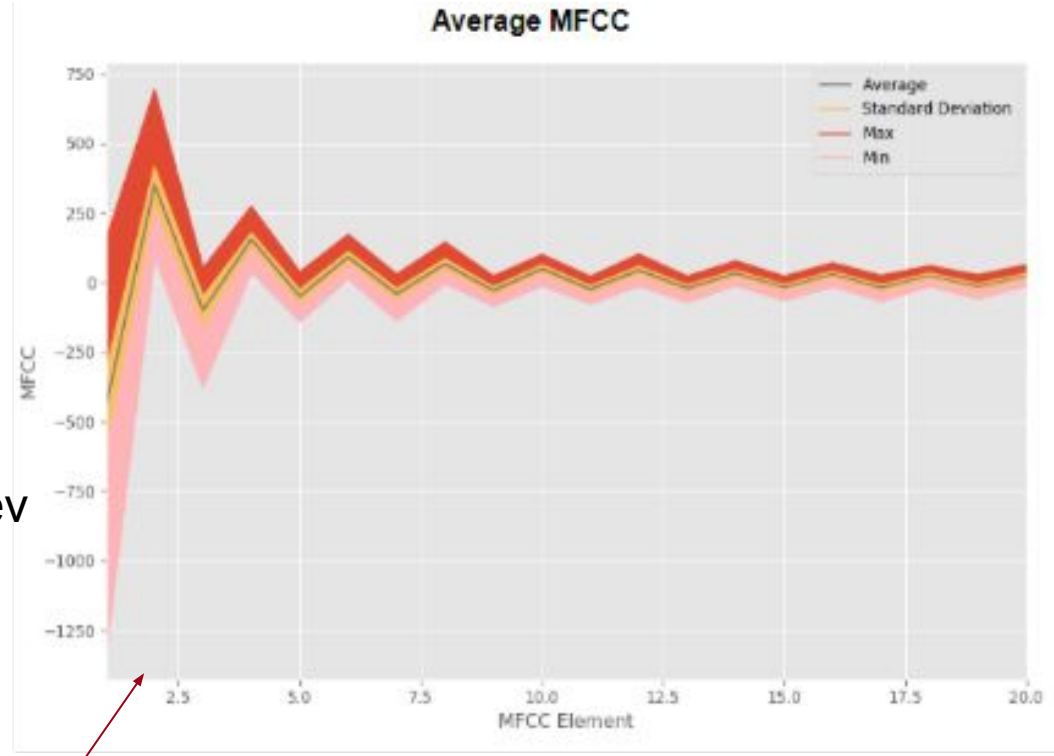
Average values

Average + standard dev

Average - standard dev

Maximum

Minimum



Machine Learning Library: plots.py

```
dict={}
for i in range(5):
    field={"field": "FBE_"+str(i)}
    ext={"extended_stats":field}
    dict[str(i+1)]=ext
agg={"aggs":dict}
res = es.search(index="ml_library_data", body=agg)

X=[]
Yavg=[]
YstdUp=[]
YstdDown=[]
Ymax=[]
Ymin=[]
x=res["aggregations"].keys()
for i in (x):
    X.append(int(i))
    Yavg.append(res["aggregations"][i]["avg"])
    YstdUp.append(res["aggregations"][i]["avg"]+res["aggregations"][i]["std_deviation"])
    YstdDown.append(res["aggregations"][i]["avg"]-res["aggregations"][i]["std_deviation"])
    Ymax.append(res["aggregations"][i]["max"])
    Ymin.append(res["aggregations"][i]["min"])
Avg=pd.DataFrame({"Average" : Yavg},X).sort_index()
StdeUp=pd.DataFrame({"Upper Standard Deviation" :YstdUp},X).sort_index()
StdeDown=pd.DataFrame({"Lower Standard Deviation" :YstdDown}, X).sort_index()
FBEMax=pd.DataFrame({"Max" :Ymax},X).sort_index()
FBEMin=pd.DataFrame({"Min" :Ymin},X).sort_index()
plt.figure(figsize=(9.5,6))
plt.plot(Avg, color='#777777', label='Average')
plt.plot(StdeUp, color="#FBC15E", label='Standard Deviation')
plt.plot(StdeDown, color="#FBC15E")
plt.plot(FBEMax, color='#E24A33', label='Max')
plt.plot(FBEMin, color="#FFB5B8", label='Min')

plt.fill_between(Avg.index,FBEMax["Max"],StdeUp["Upper Standard Deviation"], color="#E24A33")
plt.fill_between(Avg.index,Avg["Average"],StdeUp["Upper Standard Deviation"], color="#FBC15E")
plt.fill_between(Avg.index,Avg["Average"],FBEMin["Min"], color='#FFB5B8')
plt.fill_between(Avg.index,StdeDown["Lower Standard Deviation"],FBEMin["Min"], color="#FBC15E")
plt.xlim(1,5)
#plt.ylim(-5000,12500)
plt.yscale('log')
plt.legend()
plt.xlabel("FBE Element")
plt.ylabel("FBE (log)")
plt.tight_layout()
plt.savefig('FBE.png')
os.system('curl --insecure --user pi:APSdas123 -T FBE.png sftp://10.0.12.10/~/ml_plots/FBE.png')
plt.clf()
##########################################
```

All “FBE” values

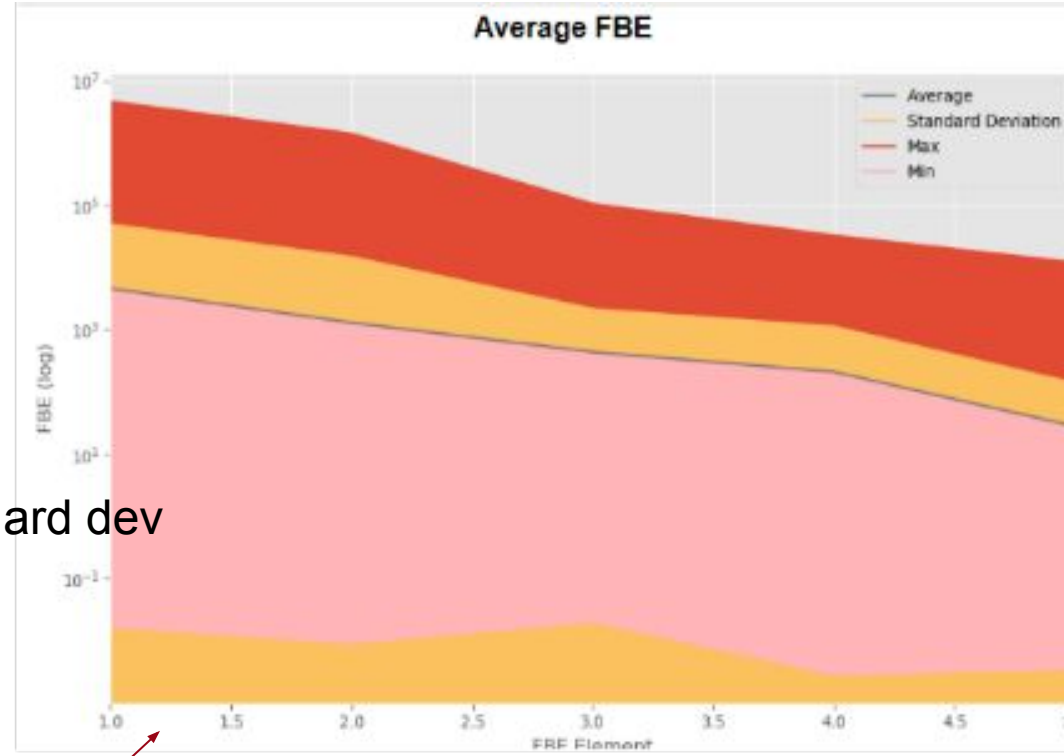
Average values

Average + standard dev

Average - standard dev

Maximum

Minimum



Machine Learning Library: plots.py

```
dict={}
for i in range(200):
    field={"field": "FFT_"+str(i)}
    ext={"extended_stats":field}
    dict[str(i+1)]=ext
agg={"aggs":dict}
res = es.search(index="ml_library_data", body=agg)

X=[]
Yavg=[]
YstdeUp=[]
YstdeDown=[]
Ymax=[]
Ymin=[]
x=res["aggregations"].keys()
for i in (x):
    X.append(int(i))
    Yavg.append(res["aggregations"][i]["avg"])
    YstdeUp.append(res["aggregations"][i]["avg"]+res["aggregations"][i]["std_deviation"])
    YstdeDown.append(res["aggregations"][i]["avg"]-res["aggregations"][i]["std_deviation"])
    Ymax.append(res["aggregations"][i]["max"])
    Ymin.append(res["aggregations"][i]["min"])
Avg=pd.DataFrame({"Average" : Yavg},X).sort_index()
StdeUp=pd.DataFrame({"Upper Standard Deviation" :YstdeUp},X).sort_index()
StdeDown=pd.DataFrame({"Lower Standard Deviation" :YstdeDown}, X).sort_index()
FFTMax=pd.DataFrame({"Max" :Ymax},X).sort_index()
FFTMin=pd.DataFrame({"Min" :Ymin},X).sort_index()
plt.figure(figsize=(9.5,6))
plt.plot(Avg, color='#777777', label='Average')
plt.plot(StdeUp, color="#FBC15E", label='Standard Deviation')
plt.plot(StdeDown, color="#FBC15E")
plt.plot(FFTMax, color='#E24A33', label='Max')
plt.plot(FFTMin, color="#FFB5B8", label='Min')

plt.fill_between(Avg.index,FFTMax["Max"],StdeUp["Upper Standard Deviation"], color="#E24A33")
plt.fill_between(Avg.index,Avg["Average"],StdeUp["Upper Standard Deviation"], color="#FBC15E")
plt.fill_between(Avg.index,Avg["Average"],FFTMin["Min"], color="#FFB5B8")
plt.fill_between(Avg.index,StdeDown["Lower Standard Deviation"],FFTMin["Min"], color="#FBC15E")
plt.xlim(1,200)
plt.ylim(-100,4000)
plt.ylabel("FFT")
plt.xlabel("FFT Element")
plt.legend()
plt.tight_layout()
plt.savefig('FFT.png')
os.system('curl --insecure --user pi:APSdas123 -T FFT.png sftp://10.0.12.10/~/ml_plots/FFT.png')
plt.clf()
```

All “FFT” values

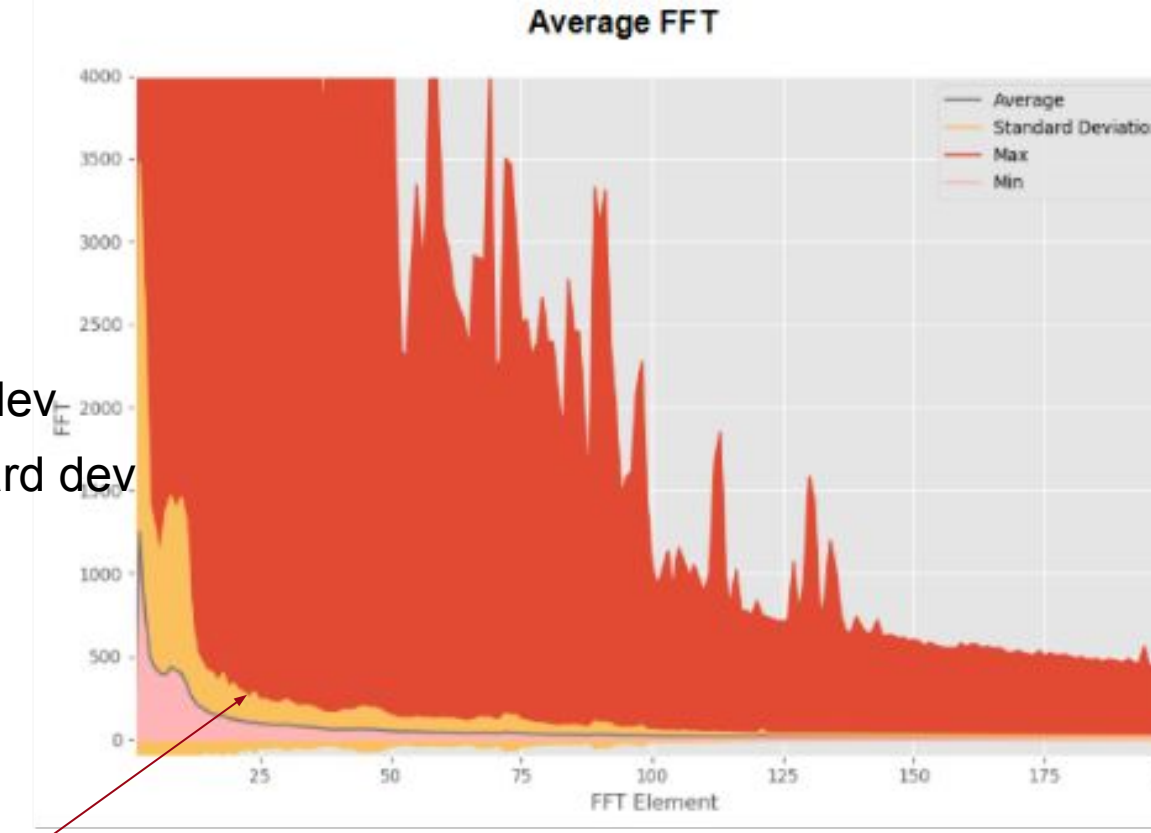
Average values

Average + standard dev

Average - standard dev

Maximum

Minimum



Machine Learning Library: plots.py

```
##################################
for classtype in cname:
    res = es.search(index="ml_library_data", body=
    #Put request in here#
    {
      "aggs": {
        "2": {
          "terms": {
            "field": "Row",
            "order": {
              "_key": "asc"
            },
            "size": 1000
          },
          "aggs": {
            "1": {
              "avg": {
                "field": "peak_to_peak"
              }
            }
          }
        }
      },
      "size": 0,
      "_source": {
        "excludes": []
      },
      "stored_fields": [
        "*"
      ],
      "script_fields": {},
      "docvalue_fields": [
        {
          "field": "uploadDate",
          "format": "date_time"
        }
      ],
###############
      "query": {
        "bool": {
          "should": [
            {
              "match_phrase": {
                "class.keyword": classtype
              }
            }
          ],
          "minimum_should_match": 1
        }
      }
#########################################
    })
    x=[]
    p2p=[]
    for i in res["aggregations"]["2"]["buckets"]:
        x.append(i["key"])
        p2p.append(i["1"]["value"])
    Normp2p=[float(i)/sum(p2p) for i in p2p]
    plt.step(x,Normp2p, label=classtype, where="mid",color="C"+str((cname.index(classtype))),linestyle=linsty[cname.index(classtype)])
#########################################################################
```

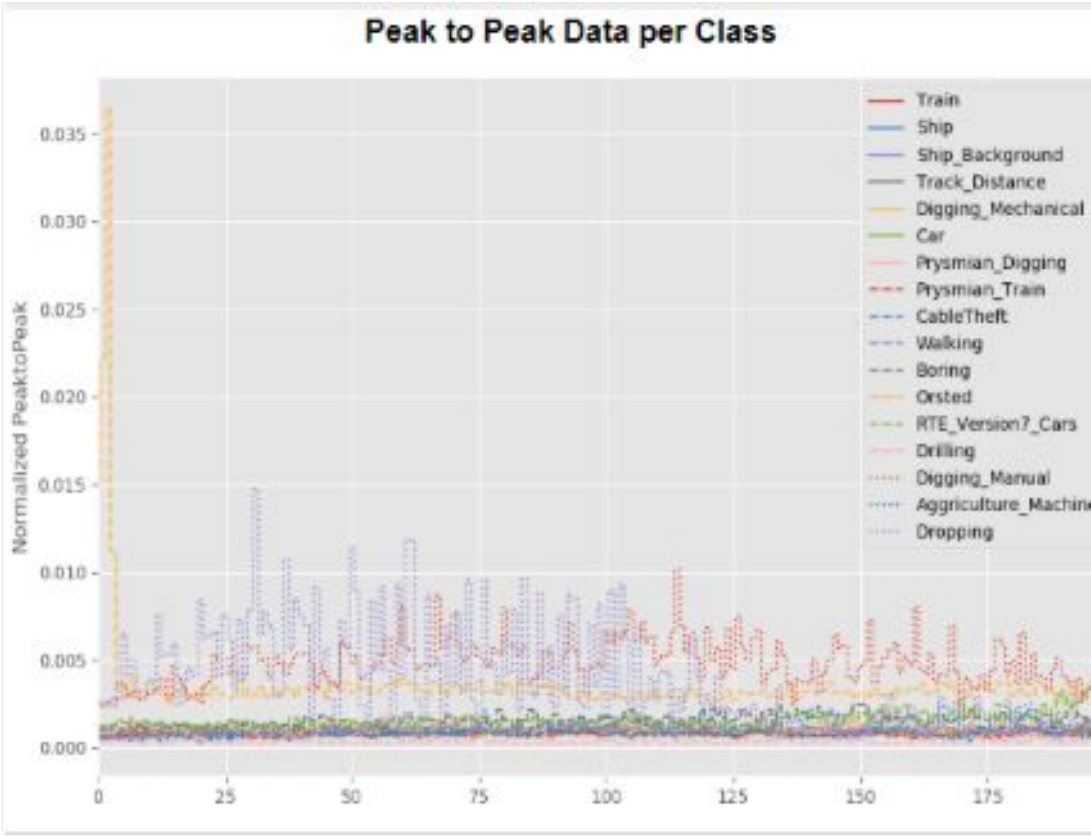
Loop over class names in cname

For each “Row” of data

“peak_to_peak”

Filter the class name

Normalize peak to peak




```
plt.xlim(0,200)
plt.ylabel("Normalized PeaktoPeak")
plt.legend()
plt.tight_layout()
plt.savefig('PeaktoPeakperClass.png')
os.system('curl --insecure --user pi:APSdas123 -T PeaktoPeakperClass.png sftp://10.8.12.10/~/ml_plots/PeaktoPeakperClass.png')
plt.clf()
#########################################################################
```

Machine Learning Library: plots.py

```
for classtype in cname:
    res = es.search(index="ml_library_data", body=
    #Put request in here#
    {
      "aggs": {
        "2": {
          "terms": {
            "field": "Row",
            "order": {
              "_key": "asc"
            },
            "size": 1000
          },
          "aggs": {
            "1": {
              "avg": {
                "field": "rms"
              }
            }
          }
        }
      },
      "size": 0,
      "_source": {
        "excludes": []
      },
      "stored_fields": [
        "*"
      ],
      "script_fields": {},
      "docvalue_fields": [
        {
          "field": "UploadDate",
          "format": "date_time"
        }
      ],
##############################
      "query": {
        "bool": {
          "should": [
            {
              "match_phrase": {
                "Class.keyword": classtype
              }
            }
          ],
          "minimum_should_match": 1
        }
      }
#########################################
    })
    x=[]
    rms=[]
    for i in res["aggregations"]["2"]["buckets"]:
        x.append(i["key"])
        rms.append(i["1"]["value"])
    Normrms=[float(i)/sum(rms) for i in rms]
    plt.step(x,Normrms, label=classtype, where="mid", color="C"+str((cname.index(classtype))),linestyle=linsty[cname.index(classtype)])
#########################################################################################
```

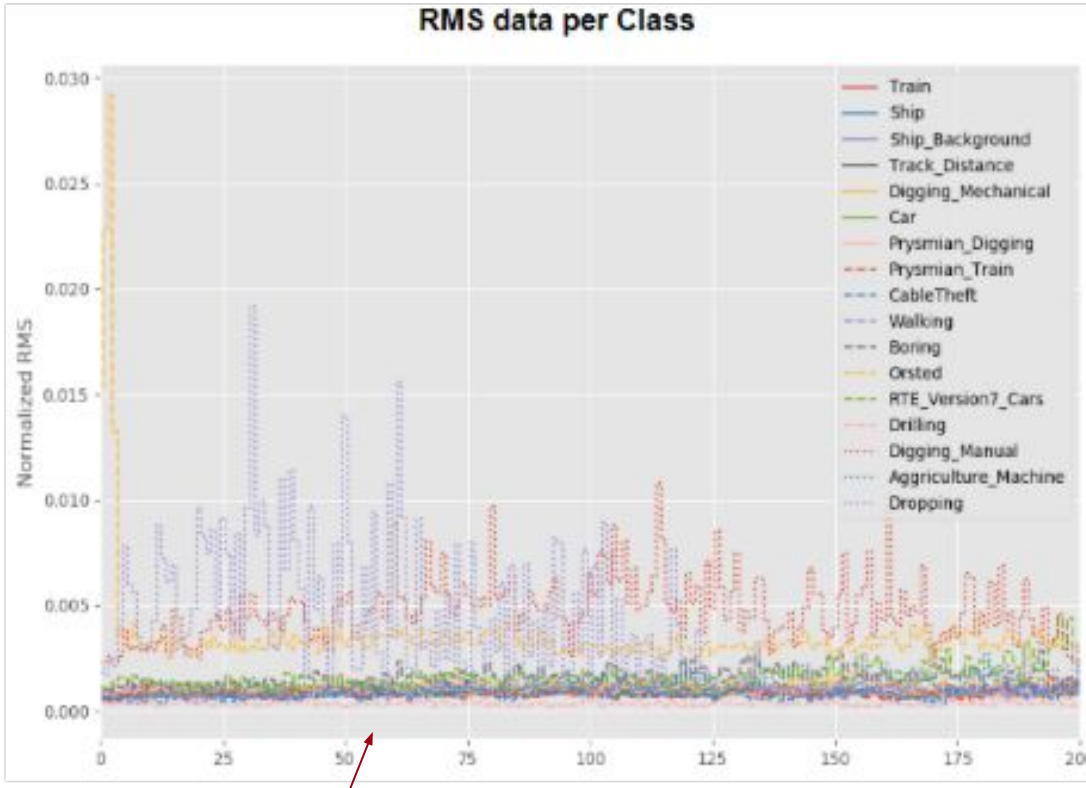
Loop over class names in cname

For each “Row” of data

“rms”

Filter the class name

Normalize rms




```
#########################################################################################
plt.ylabel("Normalized RMS")
plt.legend()
plt.xlim(0,200)
plt.tight_layout()
plt.savefig('RMSperClass.png')
os.system('curl --insecure --user pi:APSdas123 -T RMSperClass.png sftp://10.0.12.10/~/ml_plots/RMSperClass.png')
plt.clf()
```

Machine Learning Library: plots.py

```
dictc={}
for name in cname:
    match_phrase={"Class.keyword": str(name)}
    match_phrase={"match_phrase":match_phrase}
    bool={"should":[match_phrase]}
    filter={"bool":bool}
    filter=[filter]
    nbool={"must":[],"filter":filter}
    nbool={"bool":nbool}
    nfil={str(name):nbool}
    dictc[name]=nbool
dict={}
for i in range(5):
    field={"field": "FBE_"+str(i)}
    ext={"extended_stats":field}
    dict[str(i+1)]=ext
```

Class names list

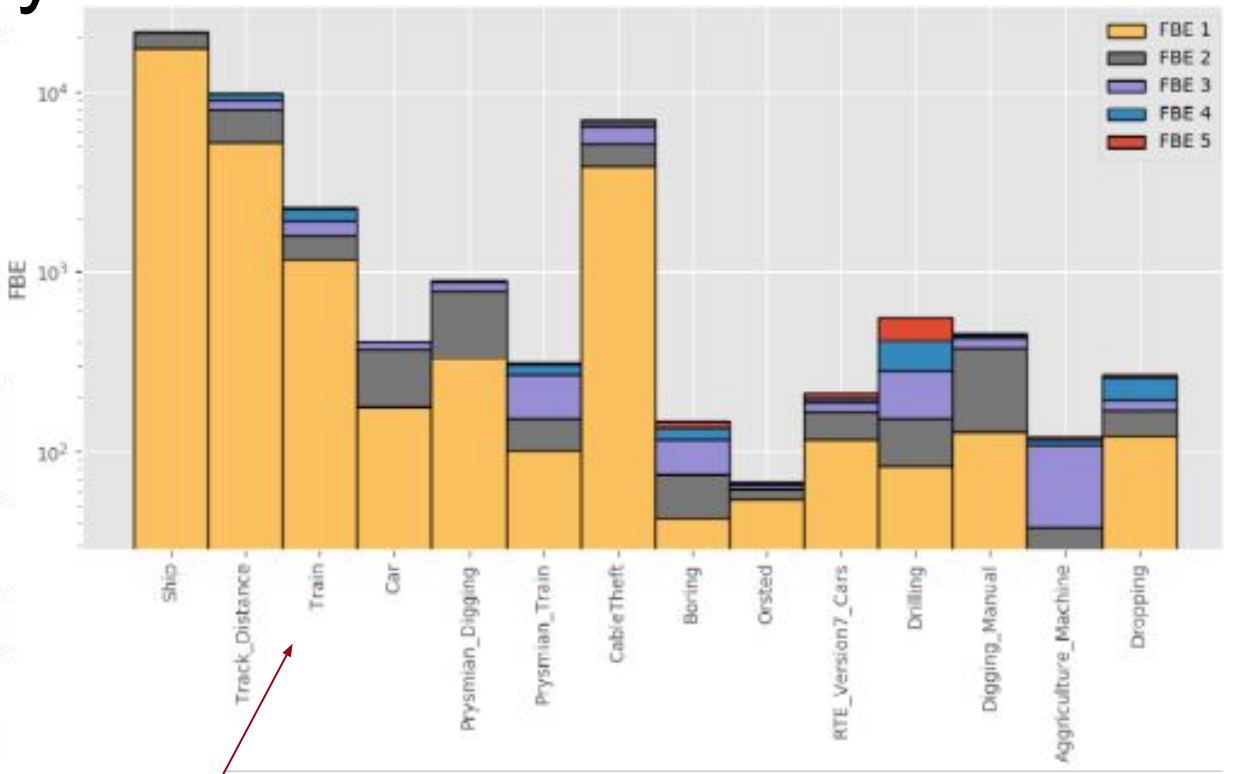
“FBE” fields

Loop over class

Class filter

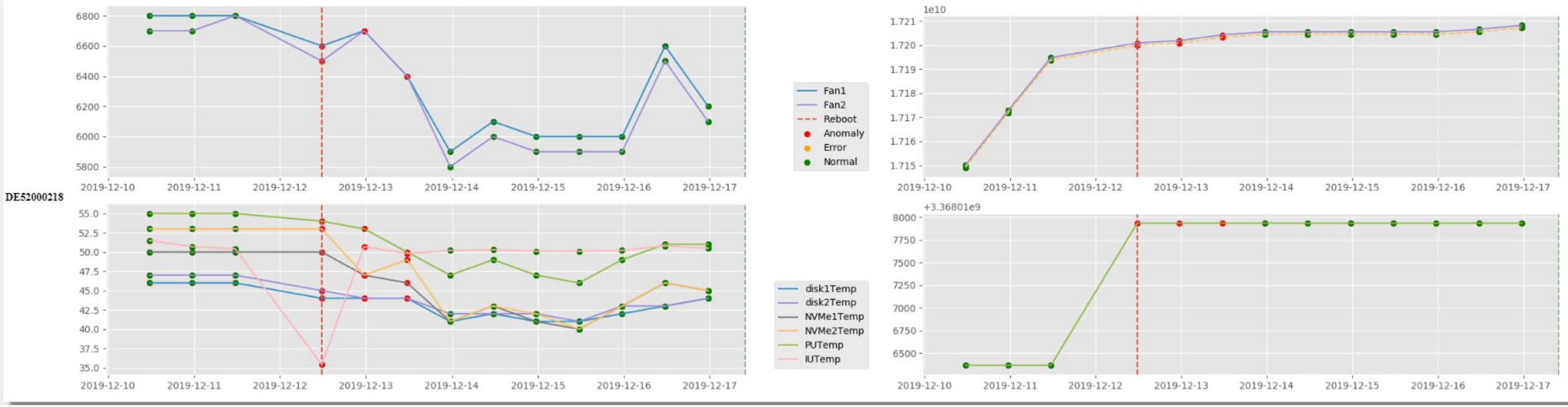
```
plt.figure(figsize=(9.5,6))
res = es.search(index="ml_library_data", body=
{"aggs": {"10": {"filters": {"filters": dictc}, "aggs":dict}}})
```

```
for i in cname:
    FB=res["aggregations"]["10"]["buckets"][i]
    plt.bar(x=cname.index(i),height=FB["1"]['avg'],color="C4",label='FBE 1', linewidth=1,edgecolor='black',width=1)
    plt.bar(x=cname.index(i),height=FB["2"]['avg'],color="C3",label='FBE 2', linewidth=1,edgecolor='black',width=1,
    plt.bar(x=cname.index(i),height=FB["3"]['avg'],color="C2",label='FBE 3', linewidth=1,edgecolor='black',width=1,
['avg'],FB["2"]['avg']).tolist())
    plt.bar(x=cname.index(i),height=FB["4"]['avg'],color="C1",label='FBE 4', linewidth=1,edgecolor='black',width=1,
bottom=np.add(np.add(FB["1"]['avg'],FB["2"]['avg']).tolist(),FB["3"]['avg']).tolist())
    plt.bar(x=cname.index(i),height=FB["5"]['avg'],color="C0",label='FBE 5', linewidth=1,edgecolor='black',width=1,
bottom=np.add(np.add(np.add(FB["1"]['avg'],FB["2"]['avg']).tolist(),FB["3"]['avg']).tolist(),FB["4"]['avg']).tolist())
plt.xticks(range(len(cname)), cname,rotation='vertical')
from collections import OrderedDict
handles, labels = plt.gca().get_legend_handles_labels()
by_label = OrderedDict(zip(labels, handles))
plt.legend(by_label.values(), by_label.keys())
plt.yscale("log")
plt.ylabel("FBE")
#plt.show()
plt.tight_layout()
plt.savefig('FBCBandsPerClass.png')
os.system('curl --insecure --user pi:APSdas123 -T FBCBandsPerClass.png sftp://10.0.12.10/~/ml_plots/FBCBandsPerClass.png')
```



Summary and Outcomes

- Explored real implementations of ML models for data science, in a workplace scenario:
  - Successfully implemented ML model to **detect anomalies** in incoming data (see plot)
  - Created **updating databases** for new **ML statistics** and **visualizations**



Summary and Outcomes

Improvements of Data Science Skills:

- **Data exploration**, **wrangling** and **analysis**, understanding and **filtering** valuable information
- Appropriate usage of different **ML models** (Linear regression, 1 class svm, autoencoders)
- Various **programming** languages and modules

Improvements of Soft Skills:

- Project **time management, task priority** assessing and keeping to **deadlines**
- **Liaising/teamwork** with colleagues on project details/plans
- Taking **leadership** on a project and reporting findings
- Tackling issues relating to project difficulties/compromises (small amounts of data, limits to software packages, etc.)
- Group meetings and scheduling
- **Creative thinking** and approaches to different data types
- **Presenting** results and project hand-off details/instructions