Sensor Analytics

AI on IoT Data

Premendra Srivastava
AI0101
Prem38sri@gmail.com
Mentor – Arpan Pal (Chief Data Scientist, TCS Innovation Labs)

Abstract: The Internet of Things (IoT) has emerged in recent years as the main technology for "Digitalization" of different Industries, be it manufacturing or energy & utilities or transportation or healthcare or smart cities. All such systems are typically characterized by a "Sense-Analyze-Respond" cycle, leading towards context-aware systems. AI and ML plays a big role in the Analyze phase. Usually the sensor data consists of time-series signals, which require a different and specialized treatment from AI-ML perspective. In this project we will look into Multi-class Classification of different sensor signals from an open dataset - a) Machine Analytics: (i) Engine noise Classification and (ii) Semi-conductor wafer fabrication abnormality detection from process sensors b) Healthcare: Disease Detection from ECG c) Human behavior: Gesture detection from Wii Remote Sensors The main objective of this project will be to understand practical issues in Sensor Data Analysis around pre-processing, feature extraction, feature selection, classifier design and tuning leading towards a framework that can be applied to a general class of similar problems. It will also help in understanding the pros-and-cons of classical ML vs. DL approaches in such scenarios.

AI on IoT Data

Rapid developments in hardware, software, and communication technologies have facilitated the emergence of Internet-connected sensory devices that provide observations and data measurements from the physical world. The technology of Internet-connected devices, referred to as Internet of Things (IoT), continues to extend the current Internet by providing connectivity and interactions between the physical and cyber worlds. These sensor based devices will generate real time data and it contains a lot of information but tapping into that data to extract useful information is a challenge that's starting to be met using the pattern-matching abilities of Artificial Intelligence. IoT devices will be among the most significant sources of new data, Artificial Intelligence will provide a considerable contribution to making IoT applications more intelligent.

So, we are applying Artificial Intelligence on real time data generated by IOT devices to get meaningful information and to make IoT devices more intelligent and useful for various use cases by applying suitable data analysis techniques and enabling them to make a decision on its own.

What is this project about?

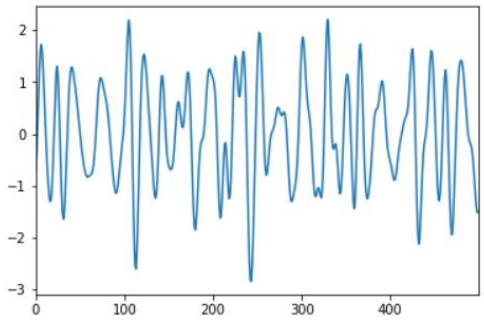
"What" of this project also defines the problem statement that we are trying to solve. Though there are different business use cases of AI on IoT but the most common one is predictive maintenance. And our goal is to build a predictive maintenance application/solution. Data generated by IOT device are time series data that varies with respect to time. There are two different dataset that we have worked on, Ford Engine dataset which is a time series data consist of engine vibration records with a label of good and bad engine and Wafer dataset which is time series dataset generated during manufacturing of semi-conductors.

And we are applying feature engineering and good practices to train and build a machine learning algorithm to solve classification problem on time series data.

How to?

How to part of our project can further be categories in below sub-parts –

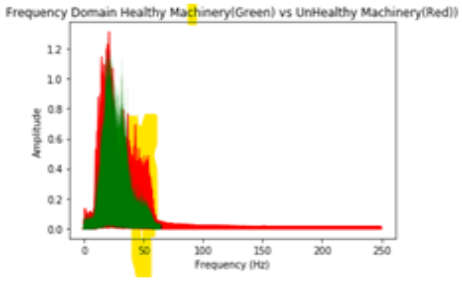
1. How to perform analysis on time series data – Performing analysis of time series signal for machine is a way different than normal data analysis approach as you don't get much insight only in time domain. Below is an snapshot of a engine data in time domain –



We have extracted statistical metrics in time domain – like (Mean, Variance, Skewness, Kurtosis, mode, Zero cut rate, Mean cut rate, RMS, Energy, Crest Factor etc.). But only this feature did not help much. Hence we started exploring different approaches to extract features from time-series data.

2. ***Feature Engineering -How to generate/extract features*** – Below are few ways to extract features from a time series data.
    a) **Fast Fourier Transformation** – Using FFT we can represent our data in

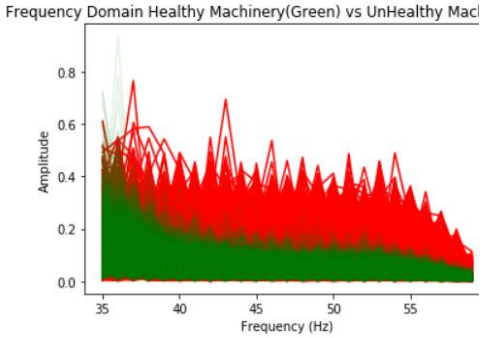
frequency domain, which show how different engines behave and achieve different frequencies –



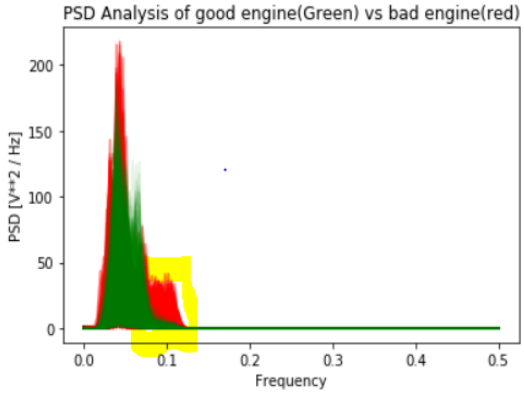
We can find amplitudes for different frequencies for engine noise and this representation in frequency domain will help us achieve features that we are using for training our classifier.

We have calculated statistical metric in FFT domain as well and added it to our Feature corpus.

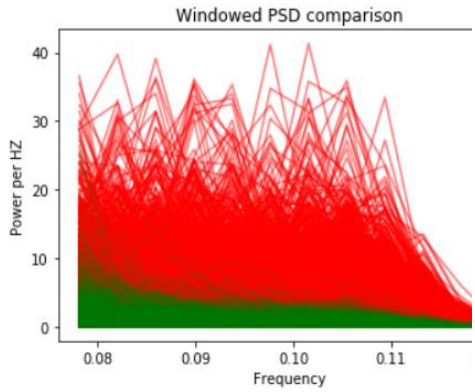
b) **Windowed FFT** – In above diagram which is representation of good engine (Green) and bad engine (red) of all 3600 engines data. We can see that highlighted section contains mostly data associated with bad engines. We can see that most variation is available during a fixed window which is 35-60 and hence we are extracting feature from this frequency window as well.



c) **Power Spectral Density** - Power spectral density function (PSD) shows the strength of the variations (energy) as a function of frequency. In other words, it shows at which frequencies variations are strong and at which frequencies variations are weak. In below diagram we can see a variation in PSD representation of good engine and bad engine. Bad engine has more peaks means more number of different frequencies and also their spread/distribution is different than good engine.

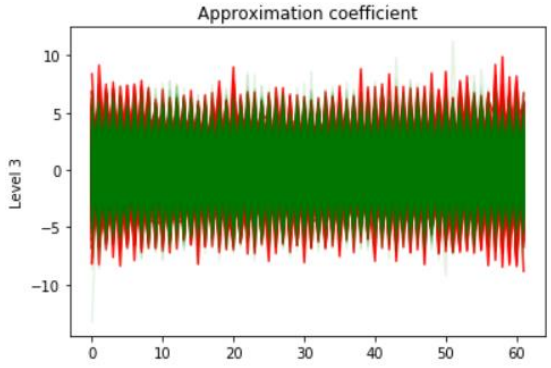


d) **Windowed PSD** – PSD representation from highlighted region will yield very good features after applying statistical metrics on data during data window. We have extracted features from this time window as well. This window covers a frequency cycle which is available mostly for bad engines.



e) **Discrete Wavelet transform** – Wavelet are small waves that exist for a finite period. We have applied discrete wavelet transformation of our signal by scaling with a wavelet and output contains information in frequency domain and also in time domain. Which help us to locate where occurrence of frequency was different? Approximation coefficient contains high frequency signal (passed from low pass filter) , Detail Coefficient contains low frequency signal(high pass filter).

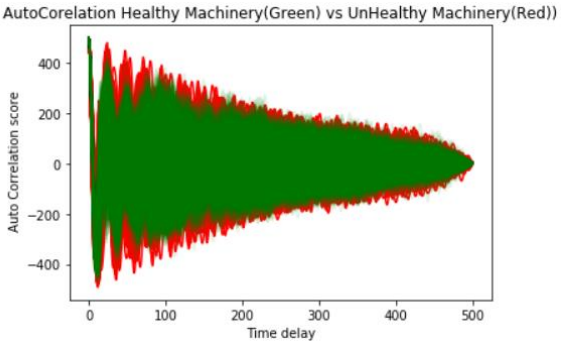
We have generated features upto 4 scales from available data post wavelet transformation.



f) **Autocorrelation** –

Autocorrelation is the correlation of a signal with a delayed copy of itself as a function of delay. Informally, it is the similarity between observations as a function of the time lag between them.

This is one very useful set of features that we will include in our feature corpus to precede further.



All of the above features (total of 596) are extracted and Exploratory Data Analysis is performed based on these features for FordA, FordB and Wafer Dataset.

3. ***Feature Engineering - How to perform feature selection*** –

Feature selection is a important part that we have dealt as understand we realized the general concept of 'curse of dimensionality'. We did selected different set of features starting from set of 10 to 60 at a step of 10 in order to reduce variance, low training time and to performance measure of the model.

We did analyzed features manually after visualizing their plot, but as the Feature corpus is big enough and we have various dataset hence we have applied automated feature selection libraries.

Below are few of feature selection techniques used by us that we have iterated from 10 to 60 features at a step of 10 to find the best features –

a) **Minimum Redundancy Maximum Relevance** –
It's a technique used to select features with minimum redundancy by correlation score and they should also contribute for maximum relevance.

b) **Recursive Feature elimination** - The Recursive Feature Elimination (or RFE) works by recursively removing attributes and building a model on those attributes that remain. It uses the model accuracy to identify which attributes (and combination of attributes) contribute the most to predicting the target attribute.

c) **Tree based feature selection** – Tree-based estimators can be used to compute feature importance's, which in turn can be used to discard irrelevant features. Technique of using ExtraTreeClassifier of ensemble library of sklearn to compute best features with high relevant importance.
With the use this technique we are able to identify set of 10 features using which our training score is over 90%.

d) GenericUnivariateSelect –
This technique select k best feature by using ANOVA ,Chi2 ,percentile test. Chi2 is not suitable in our case as our dataset contains negative data as well.

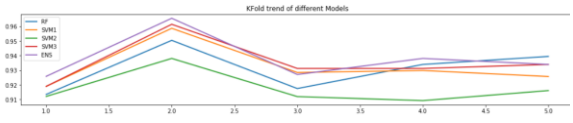
e) **Windowed Feature selection** – Selecting 3 best features from different feature windows e.g. from time domain, FFT domain, PSD etc.

f) **Union of All Best Features** –
In this feature selection technique we have selected a union of all features generated from above used feature selection techniques.

g) **Intersection of All Best Features** –
In this feature selection technique we have selected intersection of all features generated from above used feature selection techniques.

4. ***How to train a train a machine learning model*** – We have worked on 2 different classifiers, RandomForest and Support Vector Machine. Below operation are performed in order to complete training of classifiers.
   i) Read Feature Corpus in a DataFrame.
   ii) Select set of features generated from feature selection techniques only.
   iii) Apply StandardScaler on training data.
   iv) Shuffle Training data.
   v) Initiate an classifier object from sklearn library.
   vi) Perform k-Cross-Validation to check training accuracy.
   vii) Plot accuracy score.
   viii) Iterate above steps for different hyper parameter of RandomForest and SVM.

By performing above steps we are able to visually compare consistent model along with right set of parameters.



Above plot is an example where we can see that RandomForest is producing a consistent accuracy for all folds.

5. ***How to tune hyper parameters of the model*** – We are using GridSearch to iterate over set of hyper parameter for RandomForest as well as on SVM to find best hyper parameters to use.

   From this test we identified below SVM parameters which will result the best performance.

   Best SVM parameters –

```
clf_svm_rfe = svm.SVC(C=0.1, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=1, kernel='linear',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

   Best RandomForest parameters –

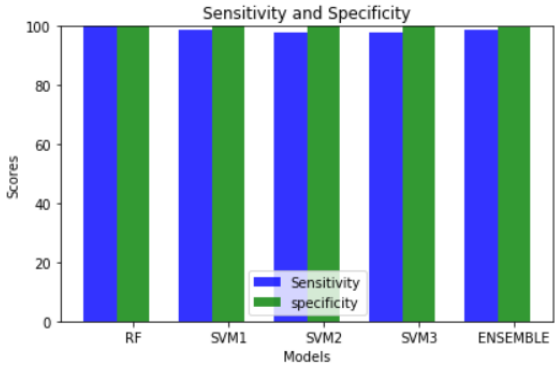
```
RFC_METRIC = 'entropy'  #metric used for RandomForrestClassifier
NUM_ESTIMATORS = 200 #number of estimators used for RandomForrestClassifier
NO_JOBS = -1 #number of parallel jobs used for RandomForrestClassifier
clf = RandomForestClassifier(n_jobs=NO_JOBS,
                             random_state=2000,
                             criterion=RFC_METRIC,
                             n_estimators=NUM_ESTIMATORS,
                             verbose=False)
```

   After trying various combination we selected RandomForest classifier as it training accuracy was consistent for all folds. There was no over fitting and model training time was also very less.

6. ***How to measure performance*** – Performance measure for our problem was Sensitivity and Specificity.

   Sensitivity stands for percentage of bad/good engines are correctly predicted by the model.

   Specificity stands for proportion of actual negatives, which got predicted as the negative.

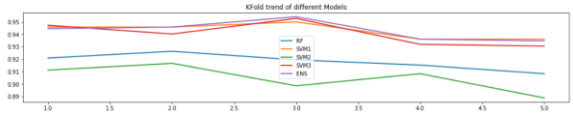


After following the complete process of feature engineering, EDA, model evaluation and performance metrics analysis we selected RandomForest classifier as it is efficient and is producing very similar result even with a set of 10 features. Such model behavior is very favorable for edge devices with less computation power.

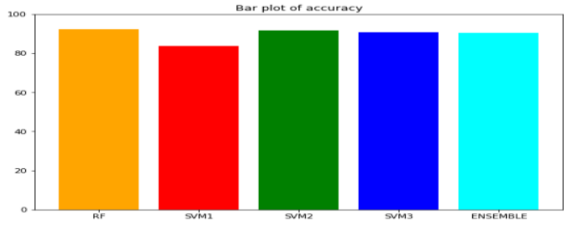
Results

Performance on all Features –

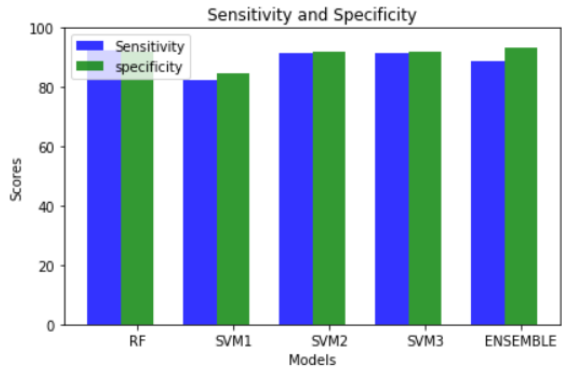
Training score –



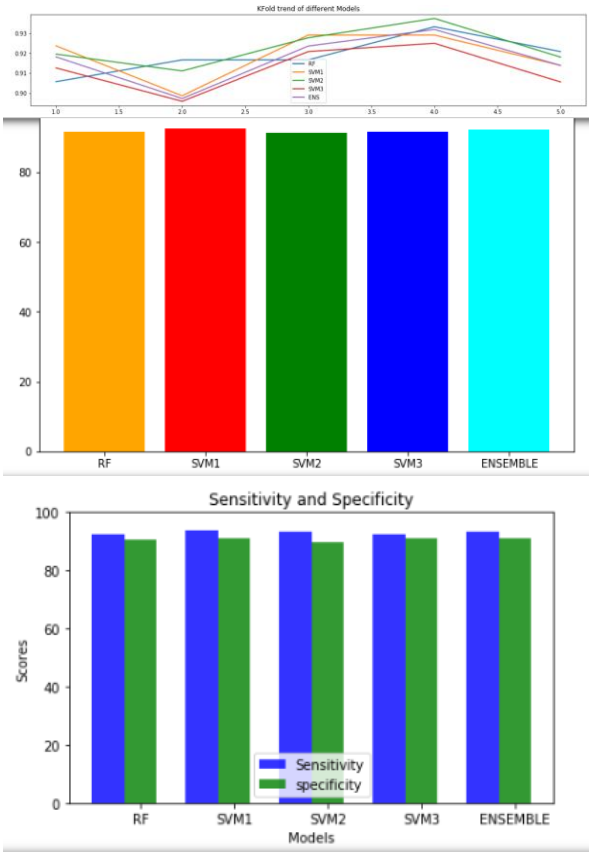
Test score –



Performance measurement –



Best performance with 10 best features -



Results for 3 different datasets

	Training accuracy	**Test Accuracy**	**Sensitivity**	**Specificity**
FordA	[0.90 0.91 0.91 0.93 0.92]	0.91	0.92	0.90
FordB	[0.91 0.92 0.93 0.94 0.93]	0.7851	0.794	0.776
Wafer	[0.99 0.99 0.99 0.99 0.99]	0.997	0.97	0.99

Note – Variation in accuracy for FordB is because training and test data are from different engine module.

- Random Forest is resulting in training accuracy upto 93-94 % and Sensitivity as 94 % with 20 features but above results are after selecting only with 10 features and a generalize approach.

Learning's

- Time series signal can be transformed to Frequency domain, wavelet, PSD etc. to get more information.
- Training data shuffle helps in consistent score across all KFold validation score.
- Models with consistent score during validation than models resulting higher value in one fold but not consistent across all.

- Curse of dimensionality – More features does not mean result in better performance of a machine learning algorithm. Hence Feature selection is one of the most important step towards building a mature machine learning model.
- Tree based feature estimators result better set of features that produces better performance.
- Features with similar correlation score should not be selected as they cause variance in test result.
- Chi2 test is applicable only for non-negative datasets.
- A better feature selection technique helps in Reduce Overfitting, Improve Accuracy and Reduce training time. Also small number of features are favorable for edge device that are built with less computational resources.
- Tuning of hyper parameters helps in improvement of Accuracy of a machine learning algorithm.
- Ensemble of stable learners is less advantageous and does not always help in increase of accuracy.
- Support Vector Machine is a a good classifier for time series datasets though training time is little more than Random Forest.
- Autocorrelation is a very good feature for time series datasets as most of the feature selection techniques contains feature from autocorrelation.

Resources/Compute

- Language – Python3.6 (Anaconda Distribution)
- Jyupiter notebook, Google Colab(MRMR feature selection, GridSearch takes lot of execution time and hence I have used Google colab with GPU runtime environment)
- Library – pandas, Numpy, Sklearn, Matplotlib, Seaborn, pymrmr, os, scipy.fftpack, scipy.signal.welch, pywt.dwt.

  Scipy.fftpack is used for coverting data to Frequency domain using FFT.
  welch from scipy.signal is used for power spectral density.
  dwt from pywt(pywavelet) is used for converting dataset to discrete wavelet transform.
  Sub-libraries like ensemble, metrics, model_selection, preprocessing, utils, svm and feature_selction is used for various purposes.
- Dataset Reference - http://www.timeseriesclassification.com/dataset.php

References

- IOT Technical Challenges and Solutions Book
- https://events.windriver.com/wrcd01/wrcm/2016/08/WP-IoT-the-internet-of-trains.pdf
- https://www.sas.com/en_in/insights/articles/big-data/machine-learning-brings-concrete-aspect-to-iot.html
- http://gwyddion.net/documentation/user-guide-en/wavelet-transform.html
- http://ataspinar.com/2018/12/21/a-guide-for-using-the-wavelet-transform-in-machine-learning/
- http://ataspinar.com/2018/04/04/machine-learning-with-signal-processing-techniques/
- https://kirankarra.wordpress.com/2018/04/29/yet-another-mrmr-implementation/
- https://www.mathworks.com/videos/understanding-wavelets-part-1-what-are-wavelets-121279.html
- https://www.investopedia.com/terms/a/autocorrelation.asp