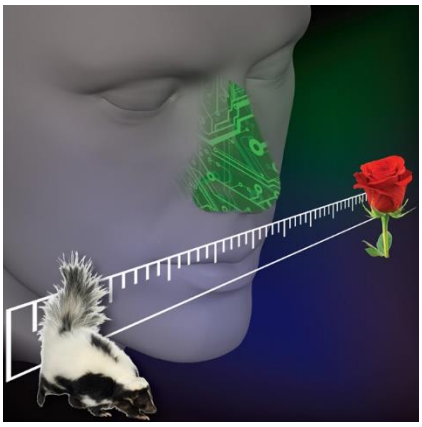
DESIGN OF E-NOSE

By-Hemanta Ingle, MS EE, San Jose State University , CA.
Guided By- Dr. Birsen Sirkeci, Professor, San Jose State University, CA
Course- Machine Learning (EE-257 Spring 2019)



MAY 13, 2019
SAN JOSE STATE UNIVERSITY
ELECTRICAL ENGINEERING DEPARTMENT

CONTENTS

1) Abstract-

Air Quality or gas sensors are largely used in electronic nose devices thanks to their high sensitivity and low cost. But for a reliable utilization of these sensors it is imperative to deal with their sensitive element alteration which is produced by gas exposures. The [6]gradual and unpredictable variation of the chemo-sensory signal responses when exposed to the same analyte under identical conditions in order to limit significantly this alteration, a very time-consuming cleaning process is necessary after each gas exposure. In this way, when continuous and rapid monitoring of a gas concentration is desired, the slow recovery process (tens of minutes) limits strongly the use of an electronic nose composed of sensors. We explore here, a Machine Learning method to reduce the impact of the gas sensor response instabilities resulted from an incomplete cleaning process of their sensitive layer. In this process we shall go through the steps like data set cleaning, visualization, developing a classification model using Logistic Regression, LDA, QDA, KNN, Support Vector Machines, Decision Tree and Random Forest.

2) Introduction-

The proposed techniques are based on the correction of sensor responses instabilities, called short-term drifts, by taking into account information coming from there errors using different models of classification. Much work has been done in the last fifteen years to develop adapted techniques and robust algorithms. The problem of data correction in presence of simultaneous sources of drift, other than sensor drift, should be also investigated since it is often the case in practical situations. To this, one idea could be combining semi-supervised methods able to learn the actual source of drift, which might clearly change with the measured samples, with adaptive drift correction strategies that can account for the continuous drift direction change in the feature space. We will go through the classification method that is having the highest accuracy and hence try to come with a model that is providing a optimal solution in prediction of the gas even if the values from the sensors deteriorate.

Keywords

Gas detectors, Cleaning, Electronic noses, Oils, Sensor phenomena and characterization, Signal processing chemical variables measurement, cleaning, compensation, electronic noses, sensor arrays, signal processing, drift impact reduction, quantitative odor analysis, metal-oxide gas sensor array, electronic nose device, cleaning process, gas concentration monitoring, slow recovery process, MOX sensor, signal processing method, pine essential oil vapor, short-term drift effect, odorant atmosphere, drift compensation, gas sensors, drift compensation, continuous monitoring, odor quantification)

3) Data Set Description-

This archive contains 13910 measurements from 16 chemical sensors utilized in simulations for drift compensation in a discrimination task of 6 gases at various levels of concentrations. The goal is to achieve good performance (or as low degradation as possible) over time, as reported in the paper mentioned below in Section 2.The dataset was gathered within January 2007 to February 2011 (36 months) in a gas delivery platform facility situated at the Chemo Signals Laboratory in the Bio Circuits Institute, University of California San Diego. Being completely operated by a fully computerized environment â€" controlled by a LabVIEW "National Instruments software on a PC fitted with the appropriate serial data acquisition boards. The measurement system platform provides versatility for obtaining the desired concentrations of the chemical substances of interest [6]with high accuracy and in a highly reproducible manner, minimizing thereby the common mistakes caused by human intervention and [6]making it possible to exclusively concentrate on the chemical sensors for compensating real drift.
The resulting dataset comprises recordings from six distinct pure gaseous substances, namely [6]Ammonia, Acetaldehyde, Acetone, Ethylene, Ethanol, and Toluene, each dosed at a wide variety of concentration values ranging from 5 to 1000 ppmv. See Tables 1 and 2 of the below cited manuscript for details on the gas identity name, concentration values, and time distribution sequence of the measurement recordings considered in this dataset.

Batch10.dat was updated on 10/14/2013 to correct some corrupted values in the last 120 lines of the file.

An extension of this dataset with the concentration values is available at Gas Sensor Array Drift Dataset at Different Concentrations Data Set

Attribute Information-

[6]Two distinct types of features were considered in the creation of this dataset:

Our [6]dataset consists of a six-gas/analyte classification problem dosed at different concentrations, in which the goal is to discriminate the six different analytes regardless of their concentration. The odor identity and concentration values in parts-per-million by volume (ppmv)

(i) The so-called steady-state feature (DR), defined as the maximal resistance change with respect to the baseline and its DR normalized version (DR divided by the acquired value when the chemical vapor is present in the test chamber)

(ii) [6]An aggregate of features reflecting the sensor dynamics of the increasing/decaying transient portion of the sensor response during the entire measurement. This aggregate of features is a transformation, borrowed from the field of econometrics and originally introduced to the chemo-sensing community by Muezzinoglu et al. (2009), that converts the [6]transient portion of the sensor response into a real scalar by estimating the maximum/minimum value y[k] for the rising/decaying portion of the exponential moving average of the sensor response:

y[k] = (1-Alfa) y[k-1]+Alfa(R[k]-R[k-1])

where R[k] is the sensor resistance measured at time k and Alfa is a scalar smoothing parameter between 0 and 1.

In particular, three different values for Alfa=0.1, 0.01, 0.001 were set to [6]obtain three different feature values from the rising portion of the sensor response and three additional features with the same Alfa values for the decaying portion of the sensor response, covering thus the entire sensor response dynamics.
thus, each feature vector contains the 8 features extracted from each particular sensor, resulting in a 128-dimensional feature vector (8 features x 16 sensors) containing all the features and organized as follows:
DR_1, |DR|_1, EMAi0.001_1, EMAi0.01_1, EMAi0.1_1, EMAd0.001_1, EMAd0.01_1, EMAd0.1_1, DR_2, |DR|_2, EMAi0.001_2, EMAi0.01_2, EMAi0.1_2, EMAd0.001_2, EMAd0.01_2, EMAd0.1_2,..., DR_16, |DR|_16, EMAi0.001_16, EMAi0.01_16, EMAi0.1_16, EMAd0.001_16, EMAd0.01_16, EMAd0.1_16
where: DR_j and |DR|_j are the R and the normalized R features, respectively. EMAi0.001_j, EMAi0.01_j, and EMAi0.1_j, are the emaR [6]of the rising transient portion of the sensor response for Alfa 0.001, 0.01, and 0.1, respectively. EMAd0.001_j, EMAd0.01_j, and EMAd0.1_j, are emaR of the decaying transient portion of the sensor response for Alfa 0.001, 0.01, and 0.1, respectively. The index j=1â€¦16 represents the number of the sensor, forming thus the 128-dimensional feature vector.

For processing purposes, the dataset is organized into ten batches, each containing the number of measurements per class and month indicated in the tables below. This reorganization of data was done to ensure having a sufficient and [6]as uniformly distributed as possible number of experiments in each batch.

Batch ID Month IDs
Batch 1 Months 1 and 2
Batch 2 Months 3, 4, 8, 9 and 10
Batch 3 Months 11, 12, and 13
Batch 4 Months 14 and 15
Batch 5 Month 16
Batch 6 Months 17, 18, 19, and 20
Batch 7 Month 21
Batch 8 Months 22 and 23
Batch 9 Months 24 and 30
Batch 10 Month 36

Batch ID: Ethanol, Ethylene, Ammonia, Acetaldehyde, Acetone, Toluene
Batch 1: 83, 30, 70, 98, 90, 74
Batch 2: 100, 109, 532, 334, 164, 5
Batch 3: 216, 240, 275, 490, 365, 0
Batch 4: 12, 30, 12, 43, 64, 0
Batch 5: 20, 46, 63, 40, 28, 0
Batch 6: 110, 29, 606, 574, 514, 467
Batch 7: 360, 744, 630, 662, 649, 568
Batch 8: 40, 33, 143, 30, 30, 18
Batch 9: 100, 75, 78, 55, 61, 101
Batch 10: 600, 600, 600, 600, 600, 600

The dataset is organized in files, each representing a different batch. Within the files, each line represents a measurement. The first character (1-6) codes the analyte, followed by the concentration level:

1: Ethanol; 2: Ethylene; 3: Ammonia; 4: Acetaldehyde; 5: Acetone; 6: Toluene

The data format follows the same coding style as in libsvm format x:v, where x stands for the feature number and v for the actual value of the feature. For example, in
1;10.000000 1:15596.162100 2:1.868245 3:2.371604 4:2.803678 5:7.512213 â€¦ 128:-2.654529

The number 1 stands for the class number (in this case Ethanol), the gas concentration level was 10ppmv, and the remaining 128 columns list the actual feature values for each measurement recording organized as described above.

4) Dataset Visualization-

[6]Our dataset consists of a six-gas/analyte classification problem dosed at different concentrations, in which the goal is to discriminate the six different analytes regardless of their concentration. The odor identity and concentration values in parts-per-million by volume (ppmv) are listed in Table 1.[6]Table 1. Analytes and concentrations in the dataset.

Analytes	Concentrations in ppmv
Ammonia	50, 60, 70, 75, 80, 90, 100, 110, 120, 125, 130, 140, 150, 160, 170, 175, 180, 190, 200, 210, 220, 225, 230, 240, 250, 260, 270, 275, 280, 290, 300, 350, 400, 450, 500, 600, 700, 750, 800, 900, 950, 1000
Acetaldehyde	5, 10, 13, 20, 25, 30, 35, 40, 45, 50, 60, 70, 75, 80, 90, 100, 120, 125, 130, 140, 150, 160, 170, 175, 180, 190, 200, 210, 220, 225, 230, 240, 250, 275, 300, 500
Acetone	12, 25, 38, 50, 60, 62, 70, 75, 80, 88, 90, 100, 110, 120, 125, 130, 140, 150, 170, 175, 180, 190, 200, 210, 220, 225, 230, 240, 250, 260, 270, 275, 280, 290, 300, 350, 400, 450, 500, 1000
Ethylene	10, 20, 25, 30, 35, 40, 50, 60, 70, 75, 90, 100, 110, 120, 125, 130, 140, 150, 160, 170, 175, 180, 190, 200, 210, 220, 225, 230, 240, 250, 275, 300
Ethanol	10, 20, 25, 30, 40, 50, 60, 70, 75, 80, 90, 100, 110, 120, 125, 130, 140, 150, 160, 170, 175, 180, 190, 200, 210, 220, 225, 230, 240, 250, 275, 500, 600
Toluene	10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100

[6]Each of the possible gas type-concentration pairs was sampled in no particular order. The resulting dataset consists of 13,910 recordings (time series sequences) collected over a period of 36 months. The exact distribution of the number of measurements per month is shown in Table 2. Notice that some of the months shown in the table do not contain any of the measurements for one or more gases; other gas analytes and/or complex mixtures that are not considered in this study were collected during this specific period of time utilizing the same array of sensors and experimental apparatus, which in turn increased the complexity of the problem due to interference from these external analytes that may potentially affect the sensor life. Additionally, as observed in Table 2, the last batch of recordings, which contains 3600 measurements from the same analytes, was purposely collected five months after the sensors had been powered off. This 5-month gap is extremely significant for this study not only because it allows us to validate our suggested method on the annotated set of measurements collected five months later, but also because it is during this period of time that the sensors were prompted to severe contamination since external interferents could easily and irreversibly get attached to the sensing layer due to the lack of the operating temperature.

[6]Table 2. Dataset details. Each row corresponds to samples collected during a period of one month for six

Month ID	Number of examples						
	Ammonia	Acetaldehyde	Acetone	Ethylene	Ethanol	Toluene	Total
month1	76	0	0	88	84	0	248
month2	7	30	70	10	6	74	197
month3	0	0	7	140	70	0	217
month4	0	4	0	170	82	5	261
month8	0	0	0	20	0	0	20
month9	0	0	0	4	11	0	15
month10	100	105	525	0	1	0	731
month11	0	0	0	146	360	0	506
month12	0	192	0	334	0	0	526
month13	216	48	275	10	5	0	554
month14	0	18	0	43	52	0	113
month15	12	12	12	0	12	0	48
month16	20	46	63	40	28	0	197
month17	0	0	0	20	0	0	20
month18	0	0	0	3	0	0	3
month19	110	29	140	100	264	9	652
month20	0	0	466	451	250	458	1625
month21	360	744	630	662	649	568	3613
month22	25	15	123	0	0	0	163
month23	15	18	20	30	30	18	131

5) Data Set Cleaning-

Data cleansing or data cleaning is the process of identifying and removing (or correcting) inaccurate records from a dataset, table, or database and refers to recognising unfinished, unreliable, inaccurate or non-relevant parts of the data and then restoring, remodeling, or removing the dirty or crude data. Data cleaning may be performed as batch processing through scripting or interactively with data wrangling tools.
After cleaning, a dataset should be uniform with other related datasets in the operation. The discrepancies identified or eliminated may have been basically caused by user entry mistakes, by corruption in storage or transmission, or by various data dictionary descriptions of similar items in various stores.
The Initial data contains irrelevant information that simply adds up to the data size and does not add on the value of the fetures.so one such kind is semicolons which in case of our data was trivial so it had to be eliminated.

6) Related Work-

Paper Topic- Chemical gas sensor drift compensation using classifier ensembles

[6]Sensor drift remains to be the most challenging problem in chemical sensing. To address this problem, we have collected an extensive dataset for six different volatile organic compounds over a period of three years under tightly controlled operating conditions using an array of 16 metal-oxide gas sensors. The recordings were made using the same sensor array and a robust gas delivery system. To the best of our knowledge, this is one of the most comprehensive datasets available for the design and development of drift compensation methods, which is freely reachable on-line. We introduced a machine learning approach, namely an ensemble of classifiers, to solve a gas discrimination problem over extended periods of time with high accuracy rates. Experiments clearly indicate the presence of drift in the sensors during the period of three years and that it degrades the performance of the classifiers. Our proposed ensemble method based on support vector machines uses a weighted combination of classifiers trained at different points of time. As our experimental results illustrate, the ensemble of classifiers is able to cope well with sensor drift and performs better than the baseline competing methods**.**

[6]Drift has plagued the sensor research community for many years, deteriorating the performance of classifiers used for gas recognition and augmenting the maintenance costs of chemo-sensory systems, or artificial electronic noses, during real-time operations. In general, sensor drift can be attributed to two predominant sources [8], [9]. First, the 'real-drift' (a.k.a. first-order drift) due to the chemical and physical interaction processes of the chemical analytes, in gas phase, occurring at the sensing film microstructure, such as aging (e.g. the reorganization of the sensor surface over long periods of time) and poisoning (e.g., irreversible binding due to external contamination). And second, the 'second-order drift' (or measurement system drift, among many other names), produced by the external and uncontrollable alterations of the experimental operating system, including, but not limited to, changes in the environment (e.g., temperature and humidity variations); measurement delivery system noise (e.g., tubes condensation, sample conditioning, etc.); and thermal and memory effects (e.g., hysteresis or remnants of previous gases). In general, a number of approaches under the notion of sensor drift counteraction have been implemented in the literature, but one of the pioneering works, and perhaps the most systematic sensor drift analysis was performed by Romain and co-workers [10], [11], who utilized a very comprehensive dataset, collected over long periods of time in real operating conditions, to provide a deep insight into the sensor drift problem, for both the real and second-order drift. Among the many interesting conclusions drawn from that work, the following three aspects were emphasized: (i) from all the sensing technologies available, metal oxide based gas sensors [12] remain the best option for long term applications for continuous monitoring systems; (ii) a calibration gas is recommended to estimate sensor drift compensation, and (iii), sensor replacement is unavoidable over long periods of time.

In practical applications, it is difficult to empirically differentiate between real drift and second-order drift, if possible at all. Accordingly, it is hard to develop methods to correct different sources of drift because the origin of it cannot be ascertained. Utilizing an effective delivery system allows the chemical sensors to bypass the second-order

drift effect, making it possible to exclusively concentrate on the chemical sensors for compensating real drift. In our particular gas delivery system, we can control the second-order drift, too, so we can exclusively address the real drift problem. Thus, in the remainder of this document we use the term drift to refer to real drift. Concerning real drift reduction, many efforts have been devoted to find the sensor materials that can reversibly interact with the gas so that the detected molecules unbind the sensor material as soon as the gas has been purged out from the sensor surface [13], [14], [6]15]. Other solutions based on periodically changing the sensor working temperature [16], [17] have also been implemented in an effort to minimize the effects of irreversibility in the sensors' responses due to poisoning. Undoubtedly, heightened reversibility in the sensor response is necessary for the effective drift counteraction. However, this general treatment only constitutes one facet of the problem—the so-called short-term drift—a substantial study of the sensor variability over longer periods of time is also necessary.

The most commonly used solutions to cope with sensor drift within the chemical sensing community are univariate and multivariate methods, where drift compensation is performed either on each sensor individually or on the entire sensor array [9], [18]. Among the multivariate drift compensation methods, unsupervised component correction techniques are the most popular [4], [19], [20]. These techniques rely on finding linear transformations that normalize the sensor responses across time so that a classifier can be directly applied to the resulting stationary data. For instance, the component correction method presented by Årtusson et al. [4] applies the following transformation to the measurement/data matrix $X \leftarrow X - (X \cdot c)c\top$, where c is the principal component vector(s) of the measurements computed using a reference gas that may approximate the drift direction. The main drawback of these techniques is that they assume the drift direction to be linear in the feature space and, therefore, a linear transformation of the data suffices to correct it. While it is entirely plausible that kernelized versions of component analysis, such as kernel principal component analysis [21], can be applied to account for non-linearities in the drift direction, these techniques have not been investigated much in the chemical sensing community. Also, with the exception of Ref. [20], these techniques require a reference gas that is used to approximate the drift direction by assuming that the reference gas provides the drift direction in all the other gases.

In this paper, we take a completely different approach to solve the mentioned problem, in which we do not make any of the abovementioned assumptions. Instead, we use a supervised machine learning method, namely, an ensemble of classifiers to cope with sensor drift. To the best of our knowledge, such a machine learning approach, that automatically detects and copes with sensor drift, has not been applied in the chemical sensing community before, although it has been shown to [6]yield promising results on problems with drifting concepts in machine learning and data mining [22], [23], [24], [25]. Utilizing a comprehensive dataset of a multi-component gas classification problem recorded from metal-oxide gas sensors over a course of 36 months, we investigate the feasibility of our ensemble of classifiers methodology to mitigate the drift effect in chemical gas sensors. It is important to note that the ensemble method used in this paper complements, rather than competes against, the existent component correction methods mentioned above; since component correction methods are essentially a pre-processing technique, we can indeed use the ensemble method on the pre-processed data, too. In the remainder of this paper, we first describe the experimental setup, the dataset, and the feature extraction methods considered in this work (Section 2). We then describe the drift compensation algorithm (Section 3), followed by a detailed description of our experimental findings (Section 4). And finally, we present the concluding comments drawn from the results presented in this paper (Section 5).

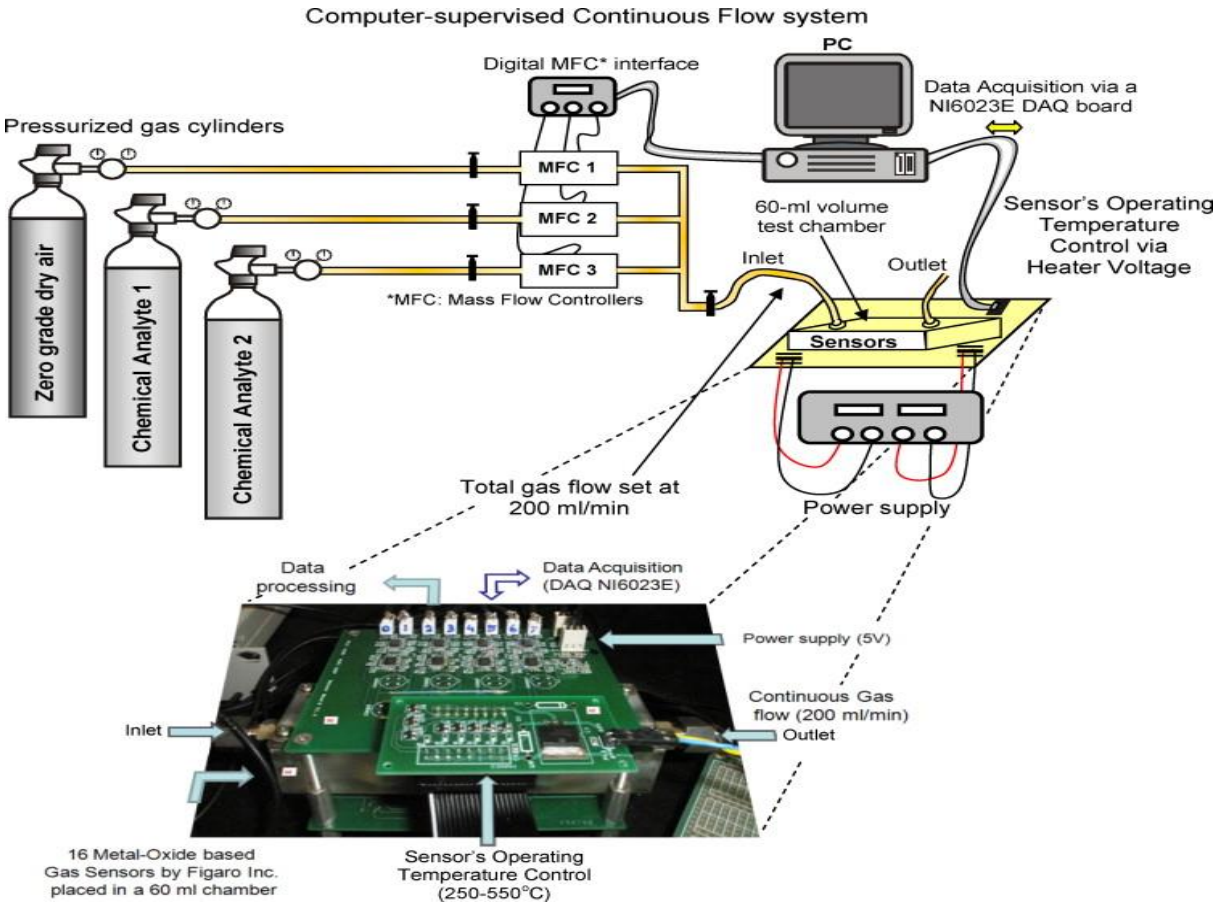
We apply our drift compensation method to an extensive dataset3 recorded by a metal-oxide gas sensor array. In this section, we describe the experimental setup, the [6]recording protocol, and the signal processing algorithms used for feature extraction.

2.1. The experimental setup

We used a sixteen screen-printed commercially available metal-oxide semiconductor gas sensors array, manufactured and commercialized by Figaro Inc. [12], for our experiments. The custom design used in the sensing technology includes an independently controlled RuO2 (Ruthenium Oxide) electrical heating line and a metal oxide semiconductor film as a sensor material printed onto the measuring electrodes (noble metal). The obtained sensor element is mounted onto an alumina substrate and then connected by lead wires to the pins of the sensor package.

The resulting array, populated by sensor devices (4 of each) tagged by the manufacturer as TGS2600, TGS2602, TGS2610, TGS2620 is placed into a 60 ml-volume test chamber, where the odorants of interest, in gaseous form, are to be injected for trials. To generate the required dataset, we connect the said test chamber to a computer-controlled continuous flow system, which provides versatility for conveying the chemical compounds of interest at the desired concentrations to the sensing chamber with high accuracy and in a highly reproducible way while keeping the total flow constant. In particular, our system utilizes three digital, computed-supervised mass flow controllers (MFCs) (provided by Bronkhorst High-Tech B.V. [26]), each of them with different maximum flow levels (200, 100, and 20 ml/min, ±1% of accuracy). Such devices connect to different pressurized gas cylinders, which contain, diluted in dry air, either the carrier gas or [6]the chemical analytes to be measured. To maintain the moisture level constant at 10% R.H. (measured at 25 ± 1 °C) during the entire measurement process, we utilize synthetic dry air as background for all measurements, provided by Airgas Inc. [27]. Then, the analytes under analysis (i.e., ammonia, acetaldehyde, acetone, ethylene, ethanol, and toluene) are added to this background in random order. The total flow rate across the sensing chamber is set to 200 ml/min and kept constant for the whole measurement process. The response of the gas sensor array was measured when the operating temperature of sensors was fixed at 400 °C, which, according to the deterministic one-to-one look-up table provided by the manufacturer [12], is attained via a built-in heater that is driven by an external DC voltage source set at 5 V. Finally, to ensure that reproducible response patterns are acquired during each measurement, the sensors were pre-heated for several days prior the experiment process gets started.

The sensor response is read-out in the form of the resistance across the active layer of each sensor; hence each measurement produces a 16-channel time series sequence. The data acquisition board collects the data from the gas sensors and controls the analog voltage signal to every sensor heater. This voltage is used to control and vary the sensor heater's operating temperature utilizing a made-in-home, LabVIEW [28] environment program running on a PC platform. The experimental setup is illustrated in the diagram shown in Fig. 1.



[6]Fig. 1. Experimental setup used for data acquisition. The sensor responses are recorded in the presence of the analyte in gaseous form diluted at different concentrations in dry air. The measurement system operates in a fully computerized environment with minimal human intervention, which provides versatility in conveying the odors of interest (at the desired concentrations) to the sensing chamber with high accuracy, and simultaneously in keeping constant the total flow. Therefore, no changes in the flow or flow dynamics are reflected in the sensor response (i.e., only the presence of an odorant will be reflected in the sensor response). Moreover, since the system continuously supplies gas to the sensing chamber (either clean dry air or a chemical component), the amount of gas molecules in the sensing chamber is homogeneously distributed.

To generate the dataset, we followed a measurement procedure consisting of the following steps. First, a constant flow of zero-grade dry air was circulated through the sensing chamber while the gas sensor array was kept at a stable operating temperature (400 °C).4 [6]This step was done to measure the baseline steady-state sensor response (i.e., the sensor response in the presence of no chemical analytes). Afterwards, the desired concentration of the odorant was injected by the continuous flow system into the sensing chamber. Finally, in the third step (cleaning phase) the vapor was vacuumed away from the sensor array and the test chamber was cleaned with dry air before the concentration phase of a new measurement. The acquisition time of these measurements took at least 300 s to complete, divided into 100 s for the gas injection phase and at least 200 s for the recovery (cleaning) phase. For processing purposes, we considered the whole sensor response after subtracting the baseline from each record. The sampling rate was set to 100 Hz. Finally, the measurement process herein described was replicated for subsequent measurements.

2.3. Data processing and feature extraction

The Figaro metal-oxide gas sensors are known to have a slow response to a chemical analyte. This response, under tightly controlled operating conditions (i.e., constant air flow and fixed operating temperature), typically involves a monotonically saturating smooth change in the conductance/resistance across its sensing layer due to the adsorption/desorption reactions of the chemical analyte occurring at the micro-porous surface of the sensor. The amount and speed of these reactions depend on (i) the analyte identity, (ii) the analyte concentration, (iii) the active layer (i.e., sensor type), and (iv) the surface temperature (i.e., the sensors' operating temperature). Since the last two factors are fixed throughout the entire measurement procedure of this analysis, the sensor-analyte identity/concentration interaction process becomes the only factor that, as a pair, shapes the response profile, and, thus, that defines the identity of the chemical analyte of interest [29]. Accordingly, features reflecting the whole sensing dynamics at the sensor surface are of special interest in our drift compensation analysis.

Feature extraction plays an important role in every chemo-sensory application [30]; it is defined as a transformation mapping the sensor response to a space of lower dimension preserving the most meaningful portion of information contained in the original sensor signal. In this work, we consider two distinct types of features that exploit the whole dynamic processes occurring at the sensor surface, including the ones that reflect its adsorption, desorption, and steady-state (or final) responses of the sensor element. On the one hand, we utilize the steady-state feature, which is the "gold-standard" for chemo-sensory feature extraction [31]. It is defined as the difference of the maximal resistance change and the baseline,

$$\Delta R = \max_k r[k] - \min_k r[k], \tag{1}$$

[6]and its normalized version expressed by the ratio of the maximal resistance and the baseline values,

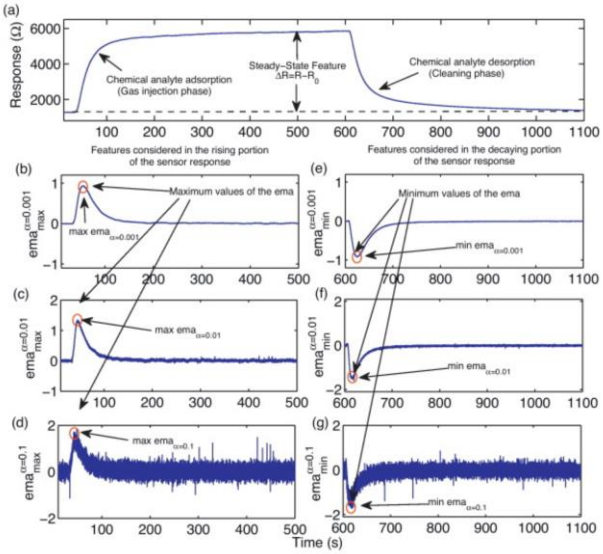
$$||\Delta R|| = \frac{\max_k r[k] - \min_k r[k]}{\min_k r[k]}, \tag{2}$$

[6]where r[k] is the time profile of sensor resistance, k is the discrete time indexing the recording interval [0, T] when the chemical vapor is present in the test chamber.

On the other hand, we also extract an aggregate of features reflecting the sensor dynamics of the increasing/decaying transient portion of the sensor response during the entire measurement procedure under controlled conditions. In particular, we utilize the exponential moving average (emaα), a transform borrowed from the field of econometrics originally introduced to the chemical sensor community in [32] that converts the increasing/decaying and saturating discrete time series r[·] collected from the chemical sensor into a real scalar fα{r[·]}, by estimating the maximum value (or minimum for the decaying portion of the sensor response) of its exponential moving average transform (emaα), calculated by,

$$y[k] = (1 - \alpha)y[k - 1] + \alpha(r[k] - r[k - 1]), \tag{3}$$

where k = 1, 2, …, T, y[[6]0], its initial condition, set to zero (y[0] = 0), and the scalar α (α ∈ {0, 1}) being a smoothing parameter of [6]the operator that defines both the quality of the feature fα{r[·]} and the time of its occurrence along the time series.5 In this work, we set three different values for α (α = 0.1, α = 0.01, and α = 0.001) as already used in our previous work [32] to obtain three different feature values starting from the pre-recorded rising portion of the sensor response and three additional features with the same α values for the decaying portion of the sensor response, covering thus the entire sensor response. Fig. 2 shows the typical signal response of a chemical sensor in the presence of 30 ppmv of Acetaldehyde and its emaα representation for the three different α values. Finally, as this figure illustrates, by applying the abovementioned transform to each of the 16 channels (sensors) in the pre-recorded time-series, we map the sensor array response into a 128-dimensional feature vector, which resulted from a combination of the 8 features described above ×16 sensors (see Table 3). For a more detailed discussion on these features, the readers are referred to Ref. [32].



[6]Fig. 2. Panel (a), typical response of a metal-oxide based chemical sensor to 30 ppmv of Acetaldehyde. The curve shows the three phases of a measurement: baseline measurement (made with pure air), test gas measurement (when the chemical analyte is injected, in gas form, to the test chamber), and recovery phase (during which the sensor again is exposed to pure air; the recovery time is usually much longer that the gas injection phase). Panels (b)–(d), exponential moving average of the rising portion of the sensor response (gas injection) for α = 0.1, α = 0.01, and α = 0.001. The maximum values of the graphs (i.e.,), [6]represent the features extracted from the chemical sensor in response to the analyte. Panels (e)–(g), exponential moving average of the decaying portion of the sensor response (cleaning phase) for α = 0.1, α = 0.01, and α = 0.001. The minimum values of the graphs (i.e.,), [6]are the features we extracted to represent the sensor signal during its cleaning phase. Features are marked with red circles in the plot. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

Table 3. Features extracted from the time series data.

Steady-state features	**Transient features**	
	Rising portion	**Decaying portion**
ΔR	$\max_k ema_{\alpha=0.001}(r[k])$	$\min_k ema_{\alpha=0.001}(r[k])$
$\|\|\Delta R\|\|$	$\max_k ema_{\alpha=0.01}(r[k])$	$\min_k ema_{\alpha=0.01}(r[k])$
	$\max_k ema_{\alpha=0.1}(r[k])$	$\min_k ema_{\alpha=0.1}(r[k])$

[6]3. Drift compensation method

We use an ensemble of classifiers [22], [24], [25] to detect and cope with sensor drift. Consider a binary classification problem with a set of features x as inputs and a class label (a gas/analyte in our problem) y as output. At every time step t, we receive a batch of examples of size mt. [6]We train a classifier ft(x), for example a support vector machine (SVM) [33], using the current batch of examples. The final classifier ht+1(x) at time step (t + 1) is a weighted combination of classifiers, i.e.,where {β1, …, βt} is the set of classifier weights. Under the assumption that the distribution of examples in the current batch St has not changed much from those in the previous batch St−1, we use the examples in batch St to estimate the weights {β1, …, βt}. [6]There are several ways to estimate these weights. A simple and intuitive way is to assign weights to classifiers according to their prediction performance on batch St. Alternatively, we could solve an optimization problem such as

$$\underset{\beta_1,\ldots,\beta_t}{\operatorname{argmin}} \sum_{i=1}^{m_t} \left(\sum_{j=1}^{t} \beta_j f_j(x_i) - y_i \right)^2. \quad (4)$$

[6]It is also possible to minimize a different loss function than the squared loss used above. For example, if the base classifiers are SVMs, we can minimize the hinge loss L(f(x), y) = max (0, 1 − yf(x)). Minimizing the squared loss as shown above has the advantage of computing the solution {β1, …, βt} [6]in closed form. The algorithm in its most general form is given below.

Algorithm 1

Algorithm to cope with concept drift

1: **for** $t = 1, \ldots, T$ **do**

2: Receive $S_t = \{(x_1, y_1), \ldots, (x_{m_t}, y_{m_t})\}$

3: Train a classifier (SVM) on S_t

4: Estimate the weights $\{\beta_1, \ldots, \beta_t\}$ using one of the techniques described in the text

5: **end for**

6: Output final classifier: $\{\beta_1, \ldots, \beta_T\}$ and $\{f_1, \ldots, f_T\}$

[6]For multiple classes, we can use one-vs-one or one-vs-all strategy [34] to train a classifier for every pair of classes or for every class respectively. Let L be the number of classes. We can estimate a set of weights for every class or c ∈ {1, …, L − [6]1} by solving (4). Alternatively, if we want to assign weights according to the prediction performance of classifiers on the most recent batch, we can simply estimate a single set of weights {β1, …, βt} using the multi-class classifier prediction accuracies on every batch. In this case, predictions are made according to a weighted majority voting, i.e.,

$$h_{t+1}(x) = \underset{y \in \{1,\ldots,L\}}{\operatorname{argmax}} \sum_{t: f_t(x)=y} \beta_t. \tag{5}$$

[6]We used this strategy in all our experiments due to its simplicity and ease of implementation. Most importantly, we can directly use the output of existing software—for instance, the widely used LibSVM [35]—to estimate the weights without the need to solve an additional optimization problem.

4. Experimental results

In all our experiments, we trained multi-class SVMs (one-vs-one strategy) with RBF kernel using the publicly available LibSVM software [35]. The features in the training and test datasets were scaled appropriately to lie between −1 and +1. The kernel bandwidth parameter γ and the SVM C parameter were chosen using 10-fold cross validation by performing a grid search in the range [2−10, 2−9, …, 24, 25] and [2−5, 2−4, …, 29, 210], [6]respectively.

We first established the fact that the sensors are drifting and that the drift is degrading the performance of classifiers. We trained a multi-class classifier on data collected during the first two months and tested it on data from the remaining months. Details on the number of measurements collected during each month for a period of three years is given in Table 2.

The classifier's performance measured in terms of prediction accuracy is shown in Fig. 3. We see that the performance gradually degrades with time, which in turn serves as a clear indicator of sensor drift and the way it has affected the accuracy of the classifier. Notice as well that on month 18 (ID: month18, in Table 2), we have only 3 measurements from class 4 (ethylene) and, therefore, the performance suddenly rises to 100%. We consider this issue to be an exception.

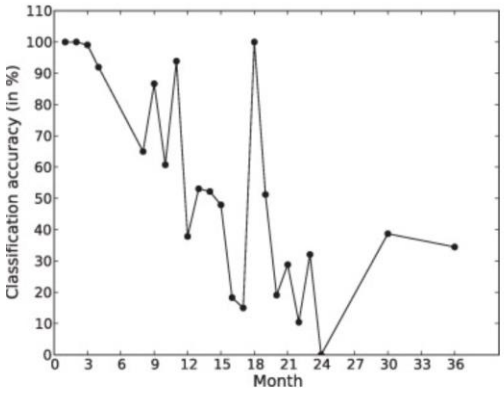


Fig. 3. Performance of the classifiers when trained on months 1 and 2 and tested on months 3–36.

[6]We then considered four settings as described below:

Setting 1 For every month, we trained a multi-class classifier with data from only the previous month and tested it on the current month.

Setting 2 For every month, we trained an ensemble of multi-class classifiers using the method in Algorithm 1.

Setting 3 Same as Setting 2 but with uniform weights on the individual (base) classifiers.

Setting 4 Same as Setting 1 but with the test sets modified using the component correction method (PCA) [4].

The SVM trained in Setting 1 is a strong baseline because it sees the most recent batch of examples which is not corrupted by drifted data from the past. If we were to train an SVM using the entire data until a time point t, then the performance would degrade even further. Note that the ensemble of classifiers trained in Setting 3 with uniform

weights essentially uses all the data from the past. We would like to emphasize that our experimental setup is robust in the sense that the baseline methods are strong. Similar experimental setup was used in Ref. [36] which was one of the first methods to address concept drift in the machine learning community. Setting 4 is similar to the experimental setup described in Refs. [19], [20].

In order to be able to train classifiers under these settings, we need sufficient number of examples in each class and month.[6]We therefore combined measurements from 36 months to form 10 batches in such a way that the number of measurements was as uniformly distributed as possible. Details on the number of measurements in each batch is given in Table 4. Fig. 4 shows the performance of an SVM (black line) trained on batch 1 and tested on batches 2–[6]10. Note that this curve is estimated with the same SVM model used in Fig. 3 but tested on data from batches instead of months.

[6]Table 4. Data set details. Each row corresponds to months that were combined to form a batch.

Batch ID	Month IDs
batch1	month1, month2
batch2	month3, month4, month8, month9, month10
batch3	month11, month12, month13
batch4	month14, month15
batch5	month16
batch6	month17, month18, month19, month20
batch7	month21
batch8	month22, month23
batch9	month24, month30
batch10	month36

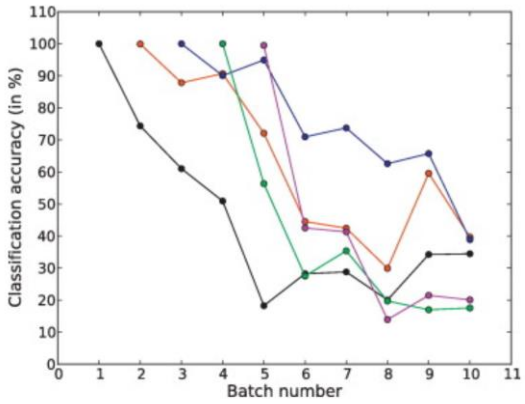


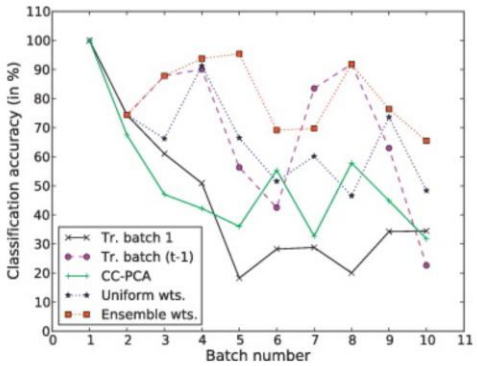
Fig. 4. Performance of the classifiers trained on batches 1–5 and tested on successive batches. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

[6]Once again, we see that the performance of the classifier is degrading with time due to drift. We found similar behaviors when we trained several SVMs on batches 2–[6]5 and tested them on successive batches. These results are again shown in Fig. 4. The complete set of results, i.e., the accuracy of classifiers trained on batches 1–9 and tested on successive batches, is given in Table 5. The first five rows in this table correspond to the results plotted in Fig. 4. These classifiers were eventually used in our ensemble method with appropriate weights. Fig. 5 shows the performance of classifiers under the settings 1, 2, 3, and 4 described above. As expected, classifiers trained under setting 1, i.e., using the most recent batch of examples, perform better than the classifier trained with data from only batch 1. We believe the drop in the accuracy when testing on batches 5 and 6 utilizing the classifiers trained on batches 4 and 5, respectively, is an artefact on the dataset. We suspect that the concentration of the gases used for training has a high impact on the performance of the classifiers over time. Accordingly, as part of an ongoing investigation, we intend to address what concentration levels have to be used to calibrate the gas sensor array. Interestingly, the classifiers trained under Setting 1 were able to cope with drift to some extent and, as mentioned above, we believe they are a natural and strong baseline for any drift-correcting machine learning algorithm to compare against.

Table 5. Performance of the classifiers trained on batches 1–9 and tested on successive batches.

Batch ID		Classification accuracy (in %) on batches 2–10							
2	3	4	5	6	7	8	9	10	
batch1	74.36	61.03	50.93	18.27	28.26	28.81	20.07	34.26	34.48
batch2		87.83	90.68	72.08	44.52	42.46	29.93	59.57	39.69
batch3			90.06	94.92	70.96	73.73	62.59	65.74	38.89
batch4				56.35	27.52	35.40	19.73	17.02	17.56

batch5	42.52	41.32	13.95	21.49	20.11
batch6		83.53	88.44	65.74	49.97
batch7			91.84	69.15	54.28
batch8				62.98	37.69
batch9					22.64

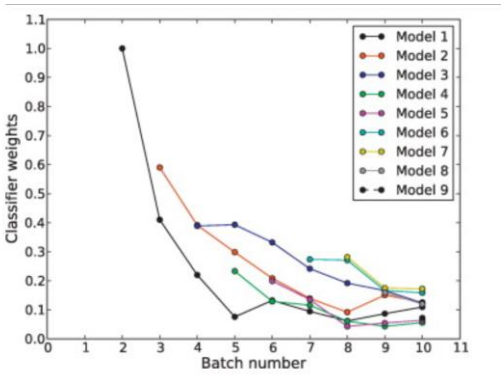


[6]Fig. 5. Performance of the classifiers trained under Setting 1 (•), Setting 2 (■), Setting 3 (★) and Setting 4 (+) described in the text. The black, continuous line corresponds to the setting where a classifier was trained with batch 1 and tested on successive batches. Setting 2 corresponds to our proposed method (Algorithm 1). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

[6]As shown in Fig. 5, the classifier ensembles were able to perform better than these baseline classifiers when tested on most of the batches with significant improvements in accuracy on several batches. From the indicated figure, we can make the following statements:

The classifier ensemble performs better than the SVM trained in Setting 1 at batches/time points t = 4, 5, 6, 9, and10. The performance is comparable at time points t = 3 and t = 8. Note that at t = 2, we do not have an ensemble and, therefore, every setting is essentially the same. The SVM in Setting 1 performs better than our approach only at t = 7 . As mentioned above, this SVM trained in Setting 1 is a very strong baseline and, thus, performing better than or as well as this setting is a positive result.

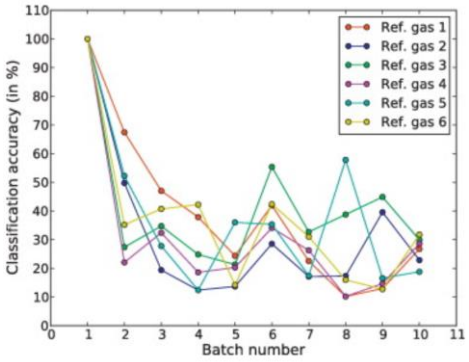
The classifier ensemble performs better than the classifier trained in Setting 3 at all the time points, i.e., from t = 3 to t = 10. This result clearly demonstrates the effectiveness of ensemble methods to automatically detect and cope with sensor drift. Fig. 6 shows how the classifier weights used in the ensemble change with time.



[6]Fig. 6. Classifier weights used in the ensembles (model 1 through model 9). At every point on the x-axis (time/batch) the corresponding points in the y-axis are the weights of the individual classifiers used in the ensemble. Notice that these weights sum up to 1 at every point of time/batch (x-axis) and that the contribution of an individual classifier to the ensemble gradually decreases [6]with time.

•

Finally, although the component correction method (Setting 4) helped to slightly improve the performance of SVM trained in Setting 1, it performed worse than all the other methods on most of the batches. Note that we applied the component correction method six times by treating every one of our six gases as a reference gas. In Fig. 5, we plot only the results obtained from the best reference gas, i.e., at every batch/time point, we apply the component correction method six times and report the best performance in terms of classifier accuracy. Fig. 7 shows the results for all the reference gases.



[6]Fig. 7. Performance of the classifier trained under Setting 4 with component correction. The individual plots correspond to the performance of classifier trained with batch 1 and tested on batches at subsequent time points after applying the component correction method for every one of the six [6]reference gases.

As a final remark, we would like to mention that our proposed method for sensor drift compensation is a supervised learning algorithm. If we have trained a classifier ensemble until a certain point in time, say T, and would like to use it for future time points, then we have to repeatedly add classifiers to the ensemble and update their weights so that it can track the drifting direction. To do this, we need labels for the data since the classifiers are trained in a supervised manner. In a more realistic scenario like real-time deployment where we may not have labels for data in future time points, then we have to update the classifier ensemble in a semi-supervised way [37]. For instance, at time T + 1, we may use the classifier ensemble trained until time T to provide labels for the new batch of data, and then we add another classifier to the ensemble using these new labels. Under the assumption that the data at time T + 1 has not

drifted much from the data at time T, this is a reasonable strategy to fill in missing labels. We leave the design and investigation of such semi-supervised learning algorithms [37] for future work.

5. Conclusions

Over a period of 3 years we have collected a comprehensive dataset using an array of 16 metal-oxide gas sensors to analyze the problem of sensor drift. The data was collected under tightly controlled conditions for 6 different gases at several concentrations ranging from 10 to 1000 ppmv. To the best of our knowledge this is one of the most comprehensive datasets available for the design and development of drift compensation methods, which will be available online as a supplementary material of this article, hopefully aiding in algorithmic improvements in the future. We have reported significant benefits using machine learning techniques based on ensembles of classifiers as a drift correction method for chemo-sensory applications. As we have demonstrated, the proposed method not only manages to mitigate well the effect of drift in chemical sensors, but also outperforms competing methods. An important observation from our sensor drift analysis is that the compensations made by the proposed method remain valid for long periods of time. The 6-month gap between the recordings from the first 30 months and the last recorded month did not invalidate the compensation of drift on the same task, despite the fact that the sensors were exposed to poisoning, as they were powered-off during this period of time.

Finally, the level of generality that the proposed drift counteraction method attains is high from the following perspective. By design, the drift counteraction method using classifier ensembles operates on samples collected on-the-fly in an online fashion which is important for real-time operations. The technique does not make any assumptions about the nature of drift, and it is also agnostic to the base classifier used in the gas identification task. As a consequence, this solution has the potential to be of use in many other applications where data is collected over an extended period of time with drifting underlying distributions, and, most importantly, it can be readily translated to more realistic gas sensing applications, including but not limited to the identification and localization of chemical analytes in complex environments (e.g., in a wind tunnel), gas distribution mapping, and gas plume tracking using robotic platforms. We held this issue as an arguable position here that we would address in further studies.

7) Developing a Model-

Developing a model is the important part of the project and here we try to analyze the parameters which classify the model to a accurate model in predicting the desired output. One of the parameter is the test error , we can note that as the complexity increases , the MSE decreases up to a point and then it starts to increase again. So the point we are looking for is that optimum point where the model doesn't overfit or underfit the output. first of all we try to explore the classification models which tune the hyperparameters and so we dive into the Random forest. While model *parameters* are learned during training—such as the slope and intercept in a linear regression—*hyperparameters* must be set by the data scientist before training. In the case of a random forest, hyperparameters include the number of decision trees in the forest and the number of features considered by each tree when splitting a node. (The parameters of a random forest are the variables and thresholds used to split each node learned during training). Scikit-Learn implements a set of sensible default hyperparameters for all models, but these are not guaranteed to be optimal for a problem. The best hyperparameters are usually impossible to determine ahead of time, and tuning a model is where machine learning turns from a science into trial-and-error based engineering.

Hyperparameter tuning relies more on experimental results than theory, and thus the best method to determine the optimal settings is to try many different combinations evaluate the performance of each model. However, evaluating each model only on the training set can lead to one of the most fundamental problems in machine learning\

```
Best Set of HyperParameters is {'criterion': 'entropy', 'n_estimators': 50
}
Accuracy of the Best Model is 0.9011746432385205
```

The n estimators =50 , gives an accuracy of 0.90 which is a good estimate .

1) CROSS VALIDATION and RANDOM FOREST

The next step is go cross validate our data and The technique of cross validation (CV) is best explained by example using the most common method, K-Fold CV. When we approach a machine learning problem, we make sure to split our data into a training and a testing set. In K-Fold CV, we further split our training set into K number of subsets, called folds. We then iteratively fit the model K times, each time training the data on K-1 of the folds and evaluating on the Kth fold (called the validation data). As an example, consider fitting a model with K = 5. The first iteration we train on the first four folds and evaluate on the fifth. The second time we train on the first, second, third, and fifth fold and evaluate on the fourth. We repeat this procedure 3 more times, each time evaluating on a different fold. At the very end of training, we average the performance on each of the folds to come up with final validation metrics for the model.

```
Accuracy for 50 trees is 90.117
Accuracy for 100 trees is 89.351
Accuracy for 200 trees is 89.234
Accuracy for 300 trees is 89.253
Accuracy for 500 trees is 89.137
```

The accuracy of the n= 50 is having a accuracy of 90.11 which sees to be higher than what we can get by incre asing the n values.

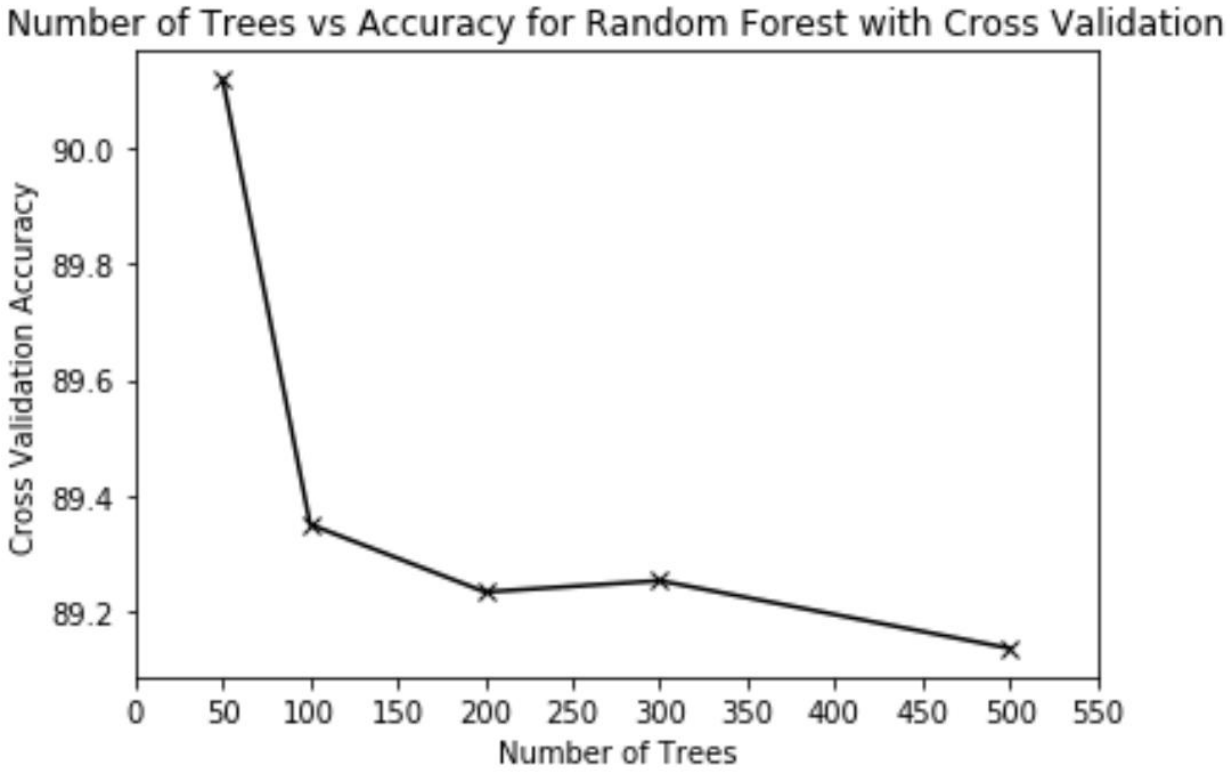


Fig-Cross validation Accuracies for different values of N

Random Forest on test dataset-

```
Number of Trees :   50  Precision : 0.628 / Recall :0.598/ Accuracy : 59.822
Number of Trees :  100  Precision : 0.655 / Recall :0.611/ Accuracy : 61.073
Number of Trees :  200  Precision : 0.656 / Recall :0.621/ Accuracy : 62.101
Number of Trees :  300  Precision : 0.663 / Recall :0.623/ Accuracy : 62.267
Number of Trees :  500  Precision : 0.651 / Recall :0.613/ Accuracy : 61.323
```

The precision and the recall can be seen from implementing the random forest of the test data
Precision is lowest for for n=50 while it is highest for n=500 and similar kind of behavior is observed with the Recal l. The accuracy of the model tends to increase with the increase in the number of trees .

Recall=True Positives / (True Positives + False Negatives)
Precision= True Positives/ (True Positives/ False Positives)

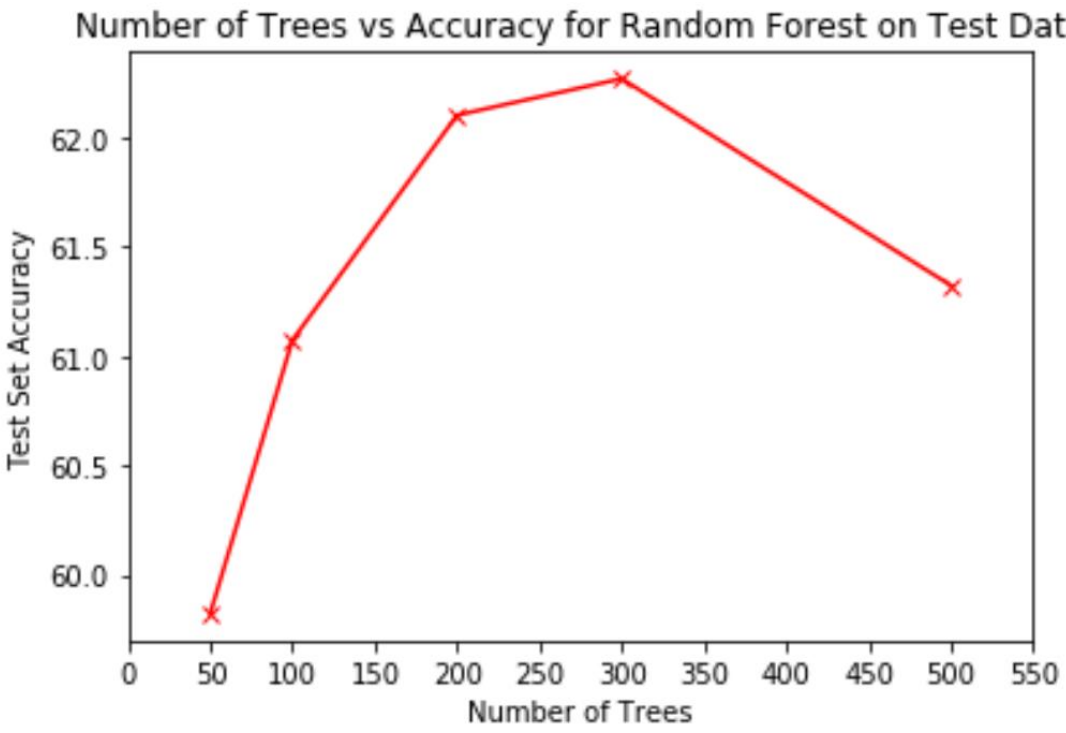


Fig-Number of trees Vs the accuracy for random forest on test data

From the above figure we can observe that the test accuracy is initially low at number of trees=50 but it gradually in creased to approximately 61 with the number of trees increased to 100 .the test set accuracy decreases rapidly form a peak point of 63 and decreases to 61.5 and goes on decreasing beyond that.

2) RANDOM FOREST WITH PCA-

```
Accuracy of the Model with Principal Components 16 is 59.906
Accuracy of the Model with Principal Components 32 is 58.794
Accuracy of the Model with Principal Components 48 is 66.796
Accuracy of the Model with Principal Components 64 is 60.072
```

Random Forest does not perform well when features are monotonic transformation of other features (this makes the trees of the forest less independent from each other). The same happens when you have more features than samples: random forest will probably overfit the dataset, and you will have a poor out of bag performance.

When using PCA you get rid of these two problems that are lowering the performance of Random Forest: you reduce the number of features. And you get rid of collinear features. (all collinear features will end up in a single PCA component, here as we can see in the output above that the accuracy of the model with principle components 16 has a less accuracy of 59.90 as compared to the one with the principle components 64 that has a accuracy of 60.1

3) RANDOM FOREST WITH CROSS VALIDATION AND PCA-

```
Accuracy for  50 trees is 90.642
Accuracy for 100 trees is 89.661
Accuracy for 200 trees is 90.982
Accuracy for 300 trees is 90.778
Accuracy for 500 trees is 90.71
```

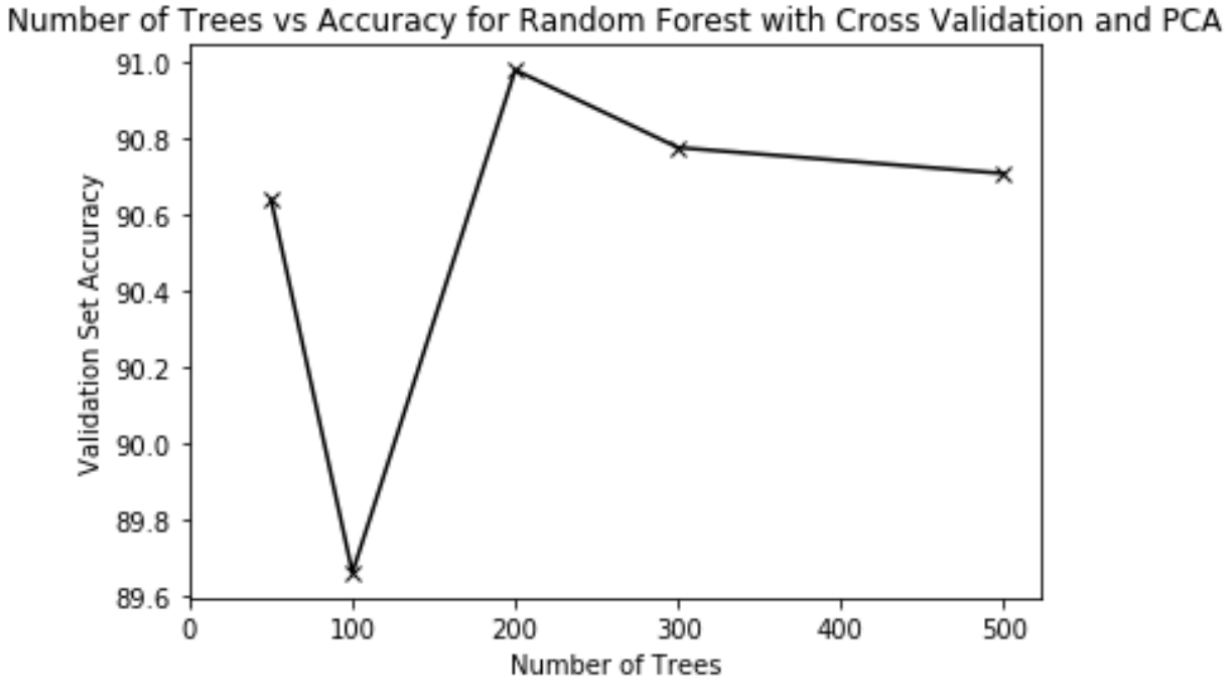


Fig -number of trees vs accuracy for random forest with cross validation

8) FINE TUNING

4) RANDOM FOREST WITH PCA ON TEST DATA SET

```
Accuracy of model with  50 trees is 64.573
Accuracy of model with 100 trees is 65.963
Accuracy of model with 200 trees is 63.684
Accuracy of model with 300 trees is 63.295
Accuracy of model with 500 trees is 63.295
```

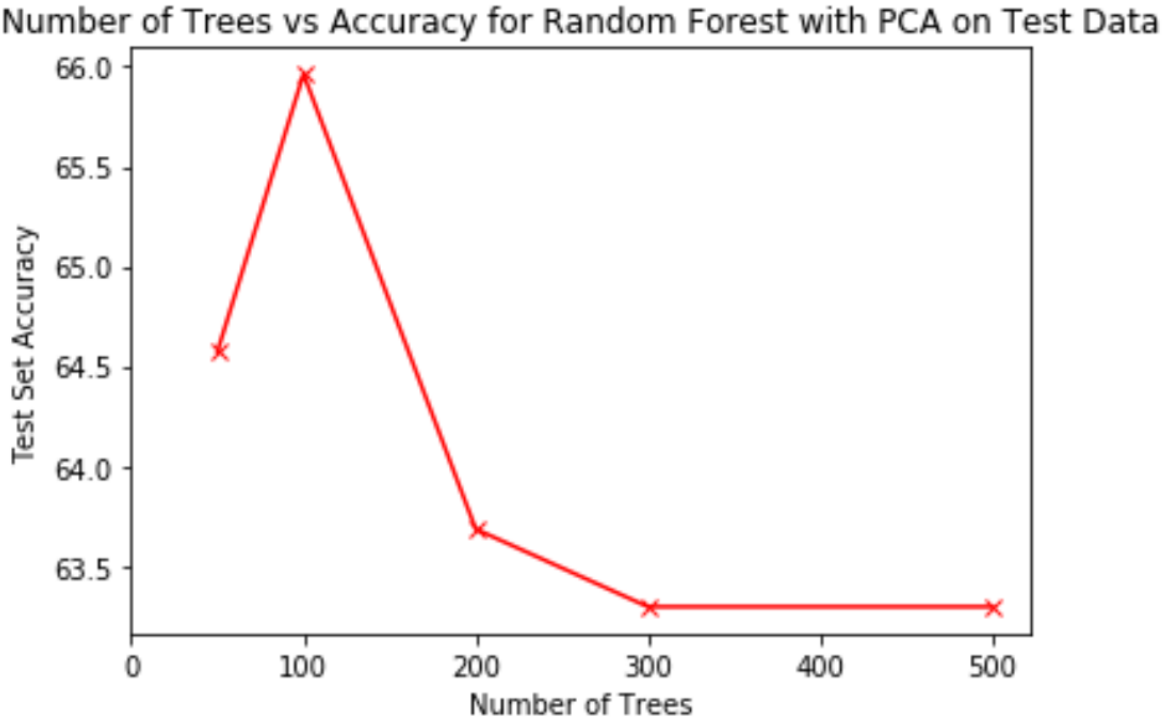


Fig-Number of Trees Vs accuracy for Random Forest with PCA on Test Data

Here we can see from the above graph that the test accuracy rapidly goes down beyond 100 number of Trees and after 300 it becomes unresponsive and come to a stagnation .

SVM WITH GRID SEARCH AND CROSS VALIDATION

```
Best Set of HyperParameters is {'C': 10, 'decision_function_shape': 'ovo', 'gamma': 'auto', 'kernel': 'rbf'}
Accuracy of the Best Model is 0.9390350451412485
```

The accuracy of the best model is 0.93.

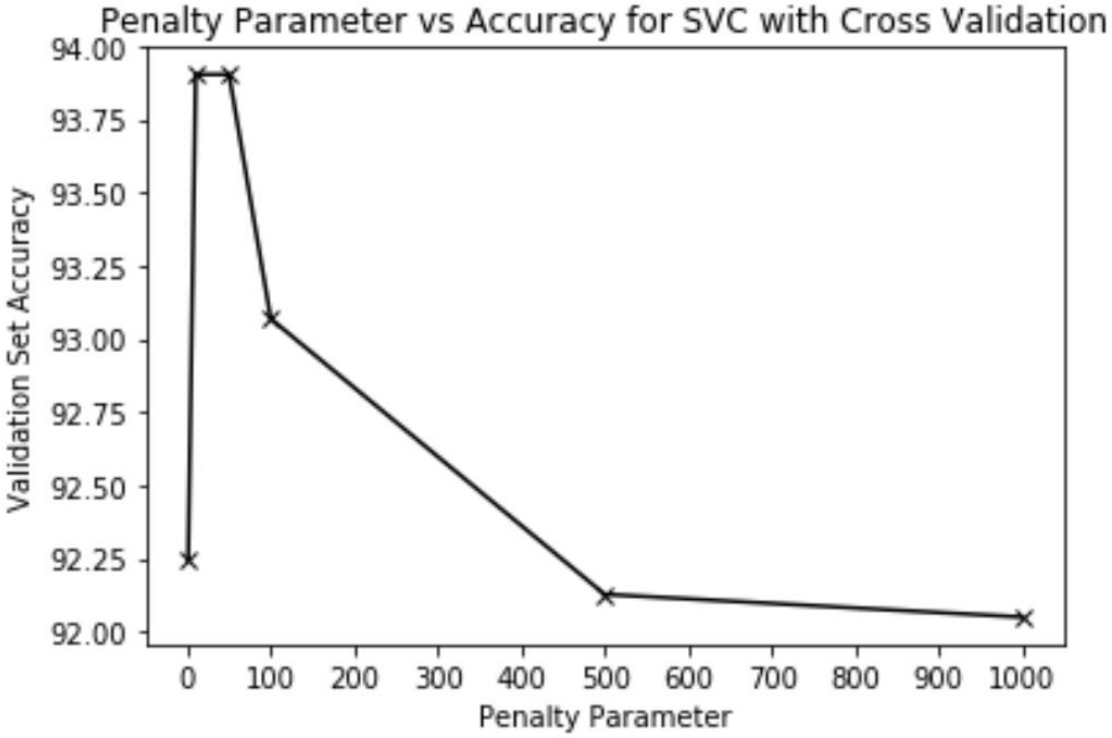


Fig-Penalty Parameter vs accuracy for SVC with Cross Validation

SVC ON TEST DATA

```
Penalty Parameter C :    1  Precision : 0.717 / Recall :0.669/ Accuracy : 66.88
Penalty Parameter C :   10  Precision : 0.795 / Recall :0.729/ Accuracy : 72.937
Penalty Parameter C :   50  Precision : 0.786 / Recall :0.719/ Accuracy : 71.853
Penalty Parameter C :  100  Precision : 0.785 / Recall :0.725/ Accuracy : 72.548
Penalty Parameter C :  500  Precision : 0.764 / Recall :0.714/ Accuracy : 71.353
Penalty Parameter C : 1000  Precision : 0.755 / Recall :0.706/ Accuracy : 70.631
```

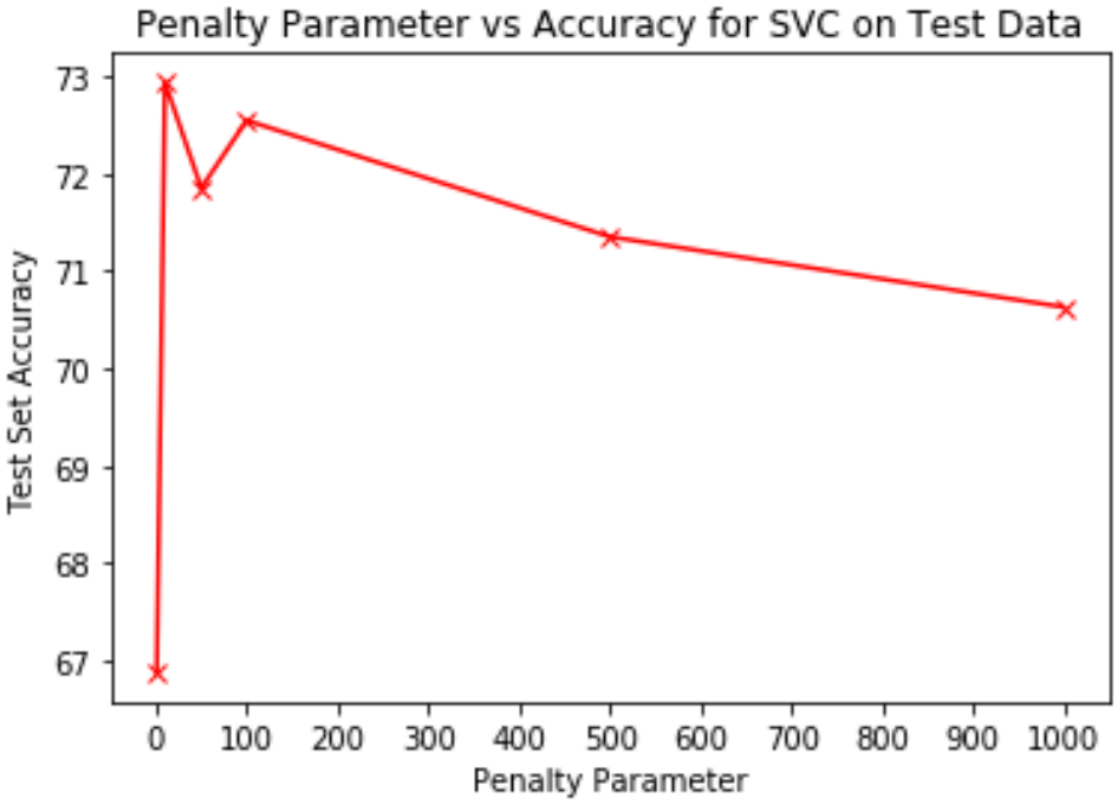


Fig Penalty Parameter vs Accuracy for SVC on Test Data

DECISION TREE CLASSIFIER WITH GRID SEARCH

```
Best Set of HyperParameters is {'criterion': 'entropy', 'splitter': 'random'}
Accuracy of the Best Model is 0.8468109892243472
```

DECISION TREE WITH CROSS VALIDATION

```
Accuracy for Decision Tree Classifier is 84.681
```

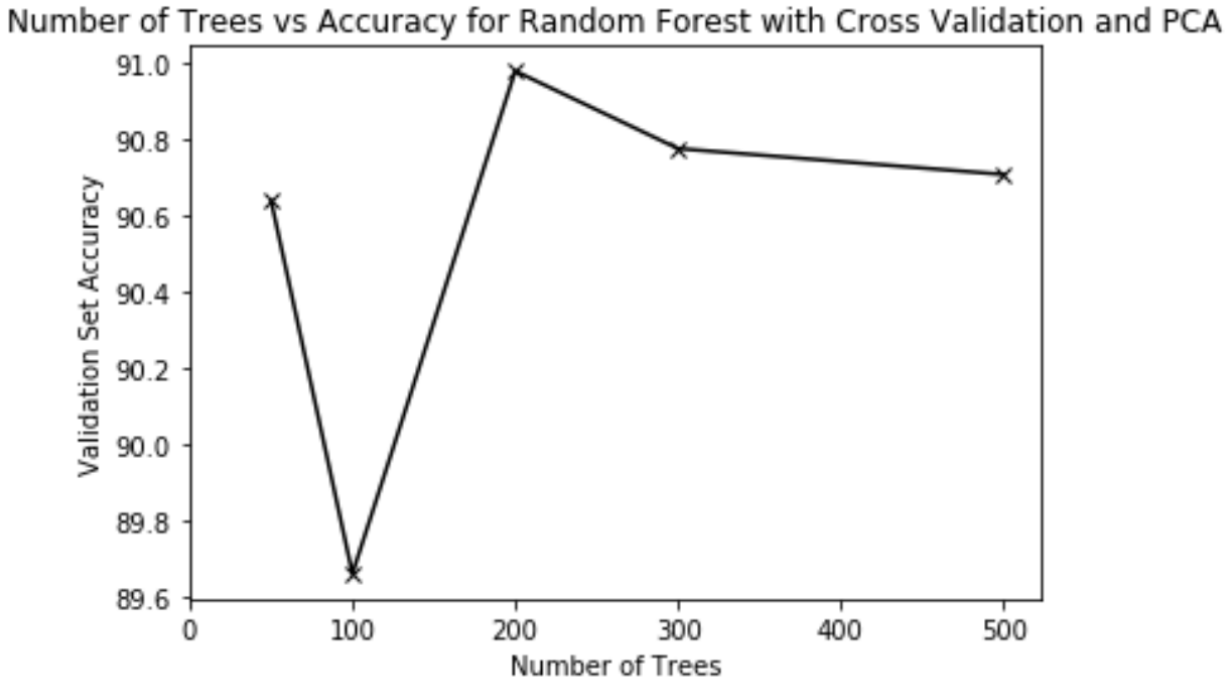
DECISION TREE ON TEST DATA

```
Precision : 0.373 / Recall :0.362/ Accuracy : 36.232
```

9) PERFORMANCE

We can see that the performance of a Random forest with PCA is found to have the highest accuracy with respect to other methods we have seen so far.

```
Accuracy for  50 trees is 90.642
Accuracy for 100 trees is 89.661
Accuracy for 200 trees is 90.982
Accuracy for 300 trees is 90.778
Accuracy for 500 trees is 90.71
```



10) NOVEL WAYS TO RESOLVE THE SENSOR DRIFT PROBLEM

Ensemble decision tree model built using bagging (bootstrap aggregation) sampling technique on gas sensor array data. Results are presented below.

Misclassification Rate - 1.1% (on test data)

Bagged Tree Confusion Matrix

	Observed Class					
Predicted Class	1	2	3	4	5	6
1	497	1	0	1	3	1
2	1	560	3	0	1	0
3	0	3	320	0	0	0
4	2	1	1	368	2	4
5	0	2	2	1	636	0
6	0	1	0	2	2	367

This is an ensemble decision tree model built using boosting technique on gas sensor array data. Adaboost algorithm was used to build the ensemble. Two variations of the model were built. In the first model Brieman coefficient was used to update the weights whereas the second model used Zhu coefficient. Results are presented below.

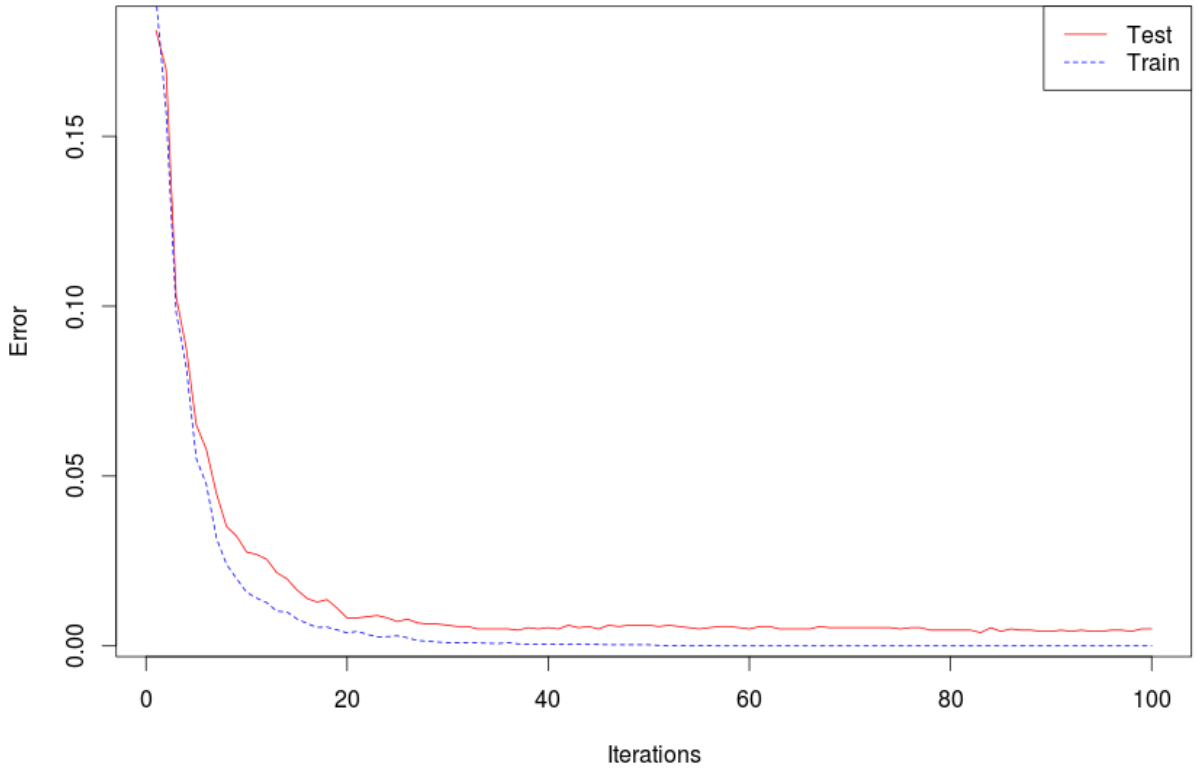
Boosted Tree Using Breiman Coefficient

Misclassification Rate - 0.5% (on test data)

Confusion Matrix

	Observed Class					
Predicted Class	1	2	3	4	5	6
1	498	1	1	0	1	0
2	1	563	0	0	0	0
3	0	1	323	0	0	0
4	1	1	0	371	1	0
5	0	2	2	0	641	0
6	0	0	0	1	1	372

Boosted Tree Error As a Function of Number of Iterations



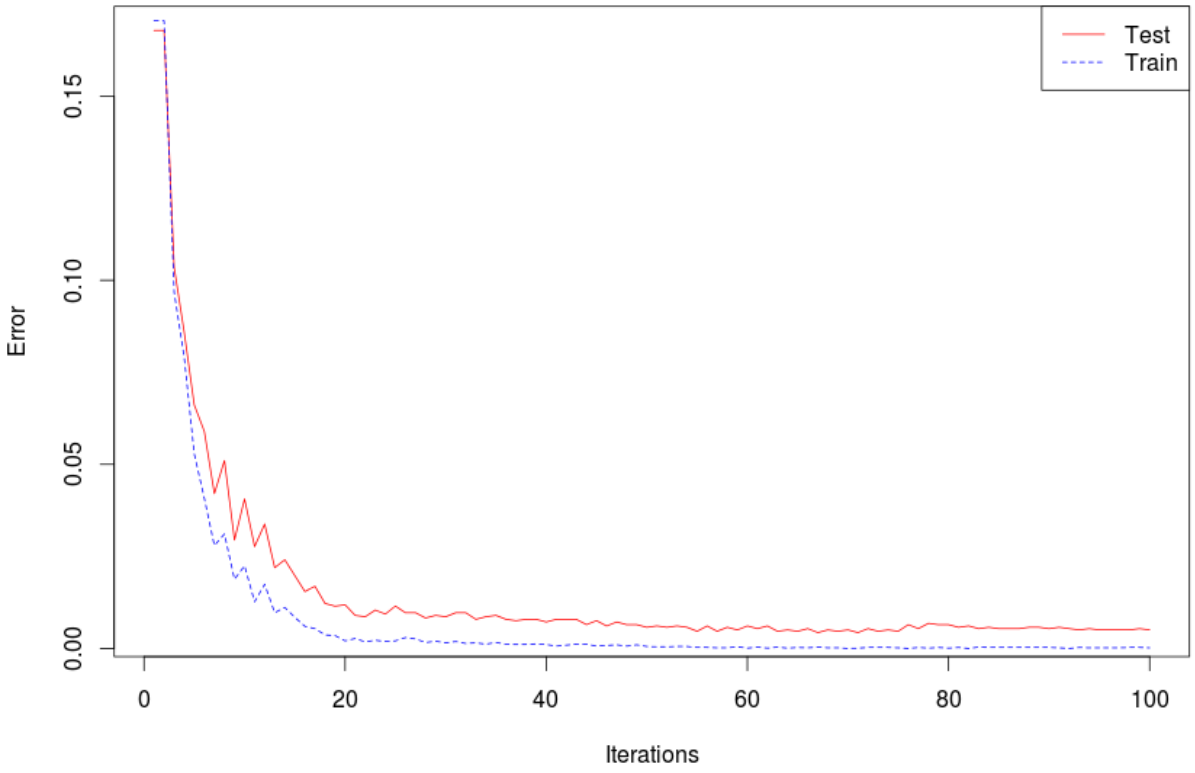
Boosted Tree Using Zhu Coefficient

Misclassification Rate - 0.5% (on test data)

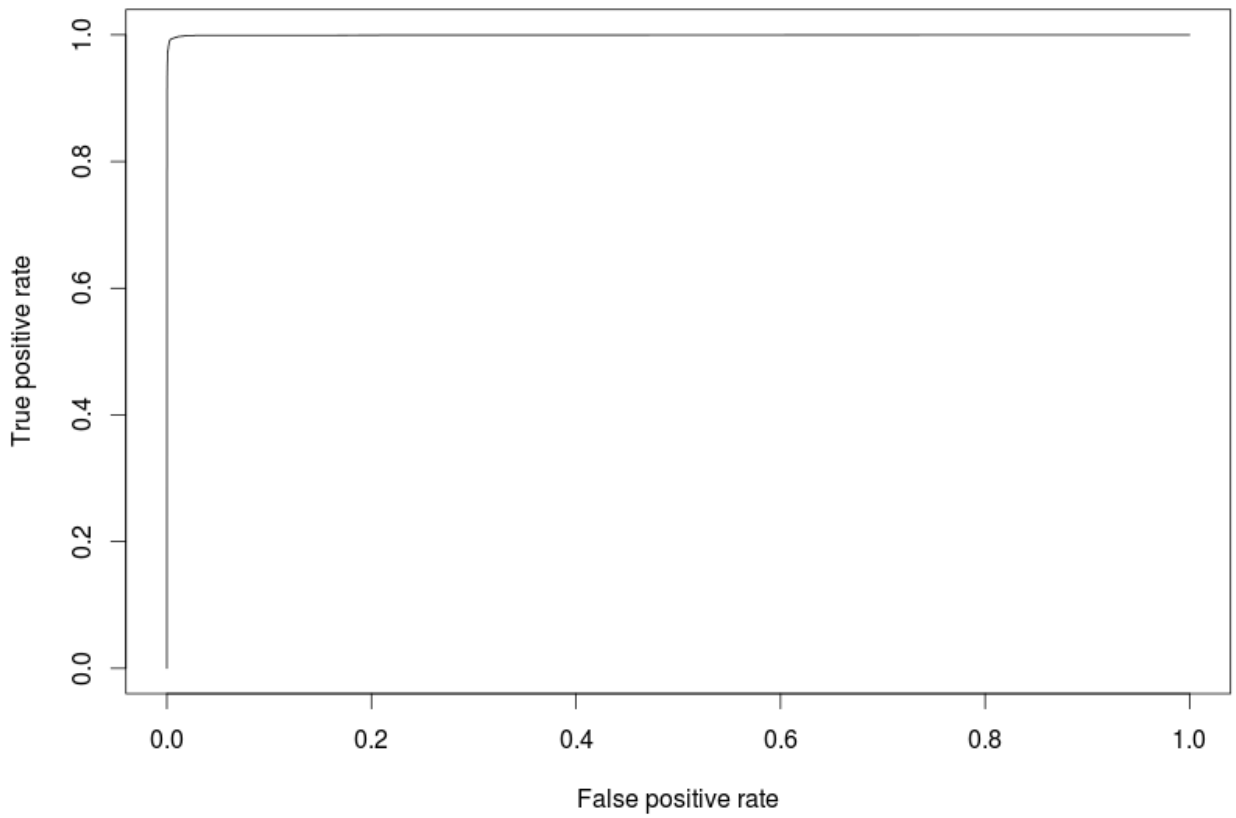
Confusion Matrix

	Observed Class					
Predicted Class	1	2	3	4	5	6
1	497	0	1	0	1	0
2	2	566	0	0	0	0
3	0	1	323	0	0	0
4	1	0	0	370	1	1
5	0	1	2	0	641	0
6	0	0	0	2	1	371

Boosted Tree Error As a Function of Number of Iterations



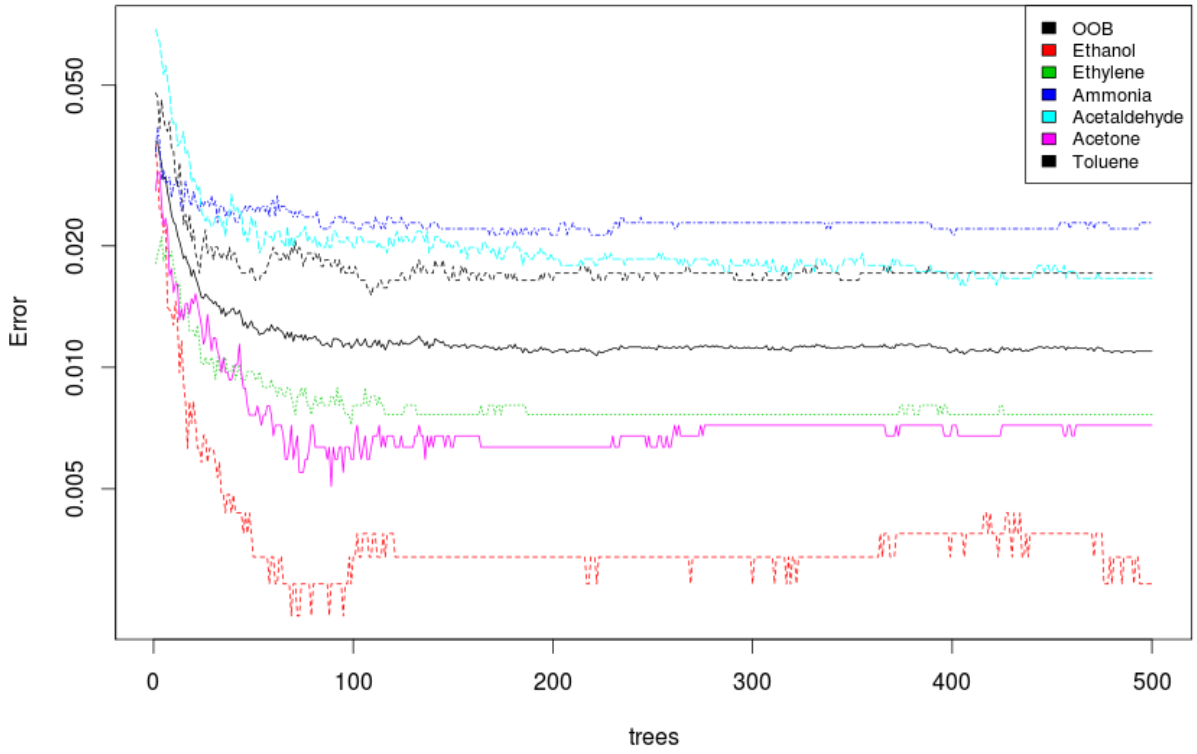
ROC Curve



Area Under Curve

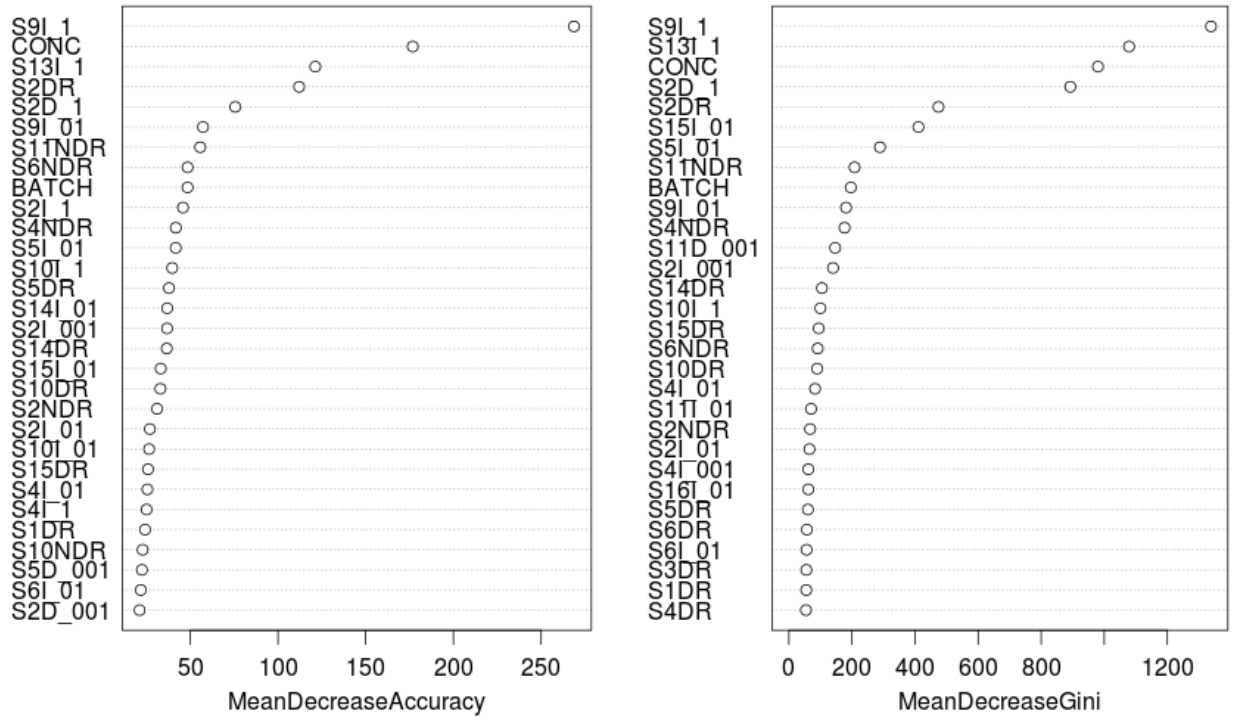
0.9995898

Bagged Tree Error As a Function of Number of Trees



OOB = Out of Bag Error

Variable Importance



Examples of how to read feature names on the above chart

S2DR - Direct Resistance of sensor 2

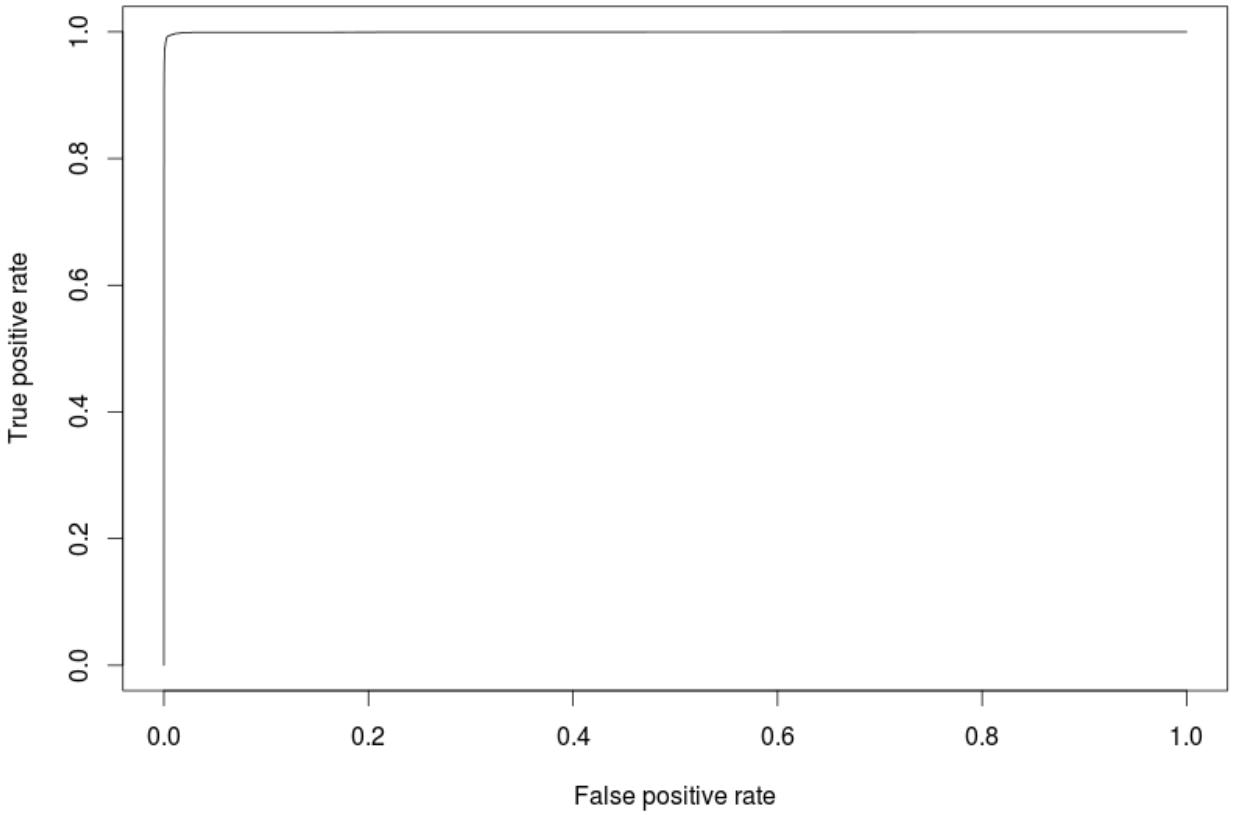
S11NDR - Normalized Direct Resistance of sensor 11

S9I_1 - Exponential Moving Average with a smoothing parameter of 0.1 when current is increasing for Sensor 9

S9D_01 - Exponential Moving Average with a smoothing parameter of 0.01 when current is decreasing for Sensor 9

CONC - Concentration (ppmv)

ROC Curve



Area Under Curve

0.9995388

Basic decision tree model on gas sensor array data. Two trees were constructed based on Entropy and Gini split criteria in R. Results are presented below.

Examples of how to read feature names on the tree

S2DR - Direct Resistance of sensor 2

S11NDR - Normalized Direct Resistance of sensor 11

S9I_1 - Exponential Moving Average with a smoothing parameter of 0.1 when current is increasing for Sensor 9

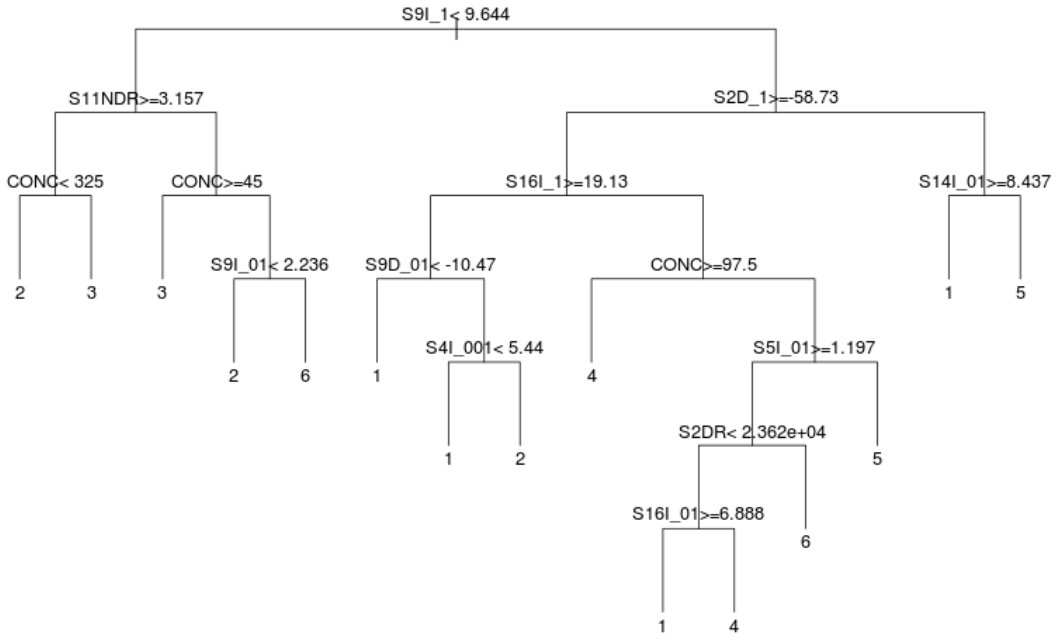
S9D_01 - Exponential Moving Average with a smoothing parameter of 0.01 when current is decreasing for Sensor 9

CONC - Concentration (ppmv)

Entropy Split Tree

Optimum Number of Splits - 14

Misclassification Rate - 16.28% (on test data)



Entropy Confusion Matrix

	Observed Class					
Predicted Class	1	2	3	4	5	6
1	379	2	0	17	5	1
2	5	531	60	0	16	0
3	6	6	255	3	8	1
4	42	14	8	280	16	36
5	38	11	2	20	587	37
6	30	4	1	52	12	297

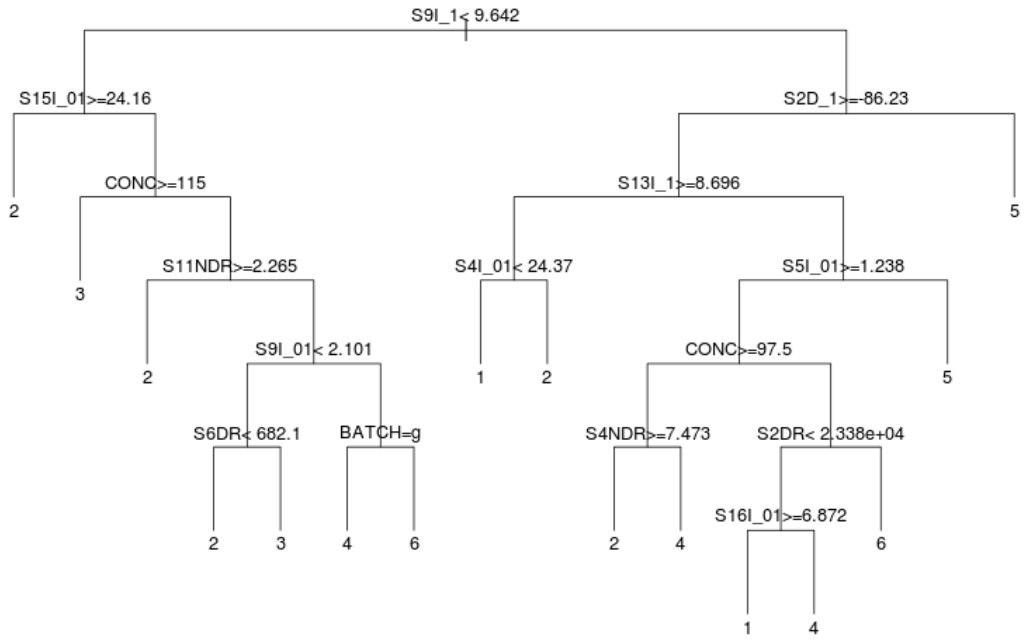
Classes - 1: Ethanol; 2: Ethylene; 3: Ammonia; 4: Acetaldehyde; 5: Acetone; 6: Toluene

---

Gini Split Tree

Optimum Number of Splits - 15

Misclassification Rate - 15.49% (on test data)

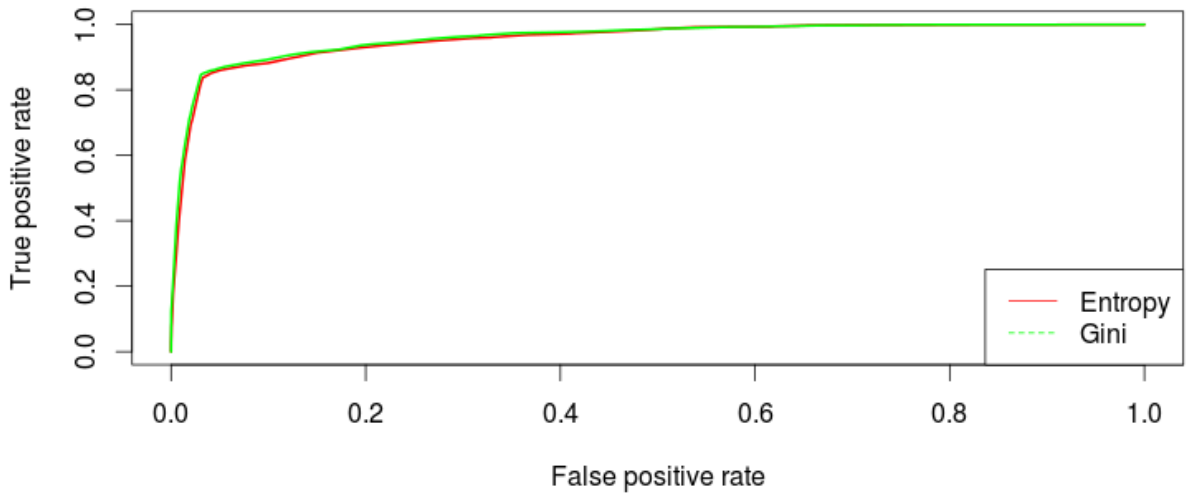


Gini Confusion Matrix

Predicted Class	Observed Class 1	2	3	4	5	6
1	203	100	29	34	67	36
2	102	295	43	61	80	26
3	28	29	132	40	49	50
4	32	51	46	127	91	55
5	85	30	15	67	324	71
6	50	63	61	43	33	134

Classes - 1: Ethanol; 2: Ethylene; 3: Ammonia; 4: Acetaldehyde; 5: Acetone; 6: Toluene

ROC Curves



Area Under Curve

Entropy Tree - 0.9548684

Gini Tree - 0.9592732

11) OVERALL DISCUSSION AND CONCLUSION-

We have tried methods classification methods to come up with a model that has high accuracy and low error rates , in context to the data of the sensors which tend to deteriorate with time a good model is the one which observes the subtle change in the concentration of the gas and then it changes its way of interpreting the data and hence we have observed till now that the Random Forest with PCA provides us the highest accuracy and the low error . more methods are to be implemented in this project in future.

12) REFERENCES-

1. Alexander I. Ivanov, Vladimir P. Kulagin, Alexey F. Kaperko, Yury M. Kuznetsov, Nataliya M. Obolyaeva, Galina M. Chulkova, Alexander N. Yurin, Alexander V. Shustrov, "Instrumental means for the control and analysis of gas mixtures based on the use of artificial neural networks", *Quality ManagementTransport and Information Security Information Technologies (IT&QM&IS) 2017 International Conference*, pp. 734-738, 2017.

2. John Leis, David Buttsworth, "A Robust Method for Tuning Photoacoustic Gas Detectors", *Industrial Electronics IEEE Transactions on*, vol. 65, no. 5, pp. 4338-4346, 2018.

3. V P. Kulagin, A. I. Ivanov, A. F. Kaperko, Y. M. Kuznetsov, N. M. Obolyaeva, G. M. Chulkova, A. N. Yurin, A. V. Shustrov, "The technology of processing information and recognizing gas mixtures using a multisensory system based on the use of neural networks", *Electronic and Networking Technologies (MWENT) 2018 Moscow Workshop on*, pp. 1-5, 2018.

4. R. Laref, E. Losson, A. Sava, K. Adjallah, M. Siadat, "A comparison between SVM and PLS for E-nose based gas concentration monitoring", *Industrial Technology (ICIT) 2018 IEEE International Conference on*, pp. 1335-1339, 2018.

5. Amine Ait Si Ali, Ali Farhat, Saqib Mohamad, Abbes Amira, Faycal Bensaali, Mohieddine Benammar, Amine Bermak, "Embedded Platform for Gas Applications Using Hardware/Software Co-Design and RFID", *Sensors Journal IEEE*, vol. 18, no. 11, pp. 4633-4642, 2018.

6. Chemical gas sensor drift compensation using classifier ensembles. (2012, March 01). Retrieved from https://www.sciencedirect.com/science/article/pii/S0925400512002018?cv=1&via=ihub

7.

Web links-

1) https://github.com/danny314/machine_learning/wiki/Bagged-Tree-Model