**Algorithms for analytes classification**

**Summary**

In this work, we have applied five algorithms to classify 40 analytes placed into 9 substrates. The simulated data consists of 18,000 spectra. Two of these algorithms are coming from the statistical field; k-nn and Logistic regression. The Support Vector Machine is from the Machine Learning area and Multiplayer Perceptron (MLP) and Convolutional Neural Networks are considered Deep Learning Algorithms. Our experimental results leads us to the conclusion that CNN algorithm has a very good performance when one particular analyte needs to be classified. The disadvantage is that has a long running time. However, reducing dimensionality by PCA decreases the CNN’s running time.

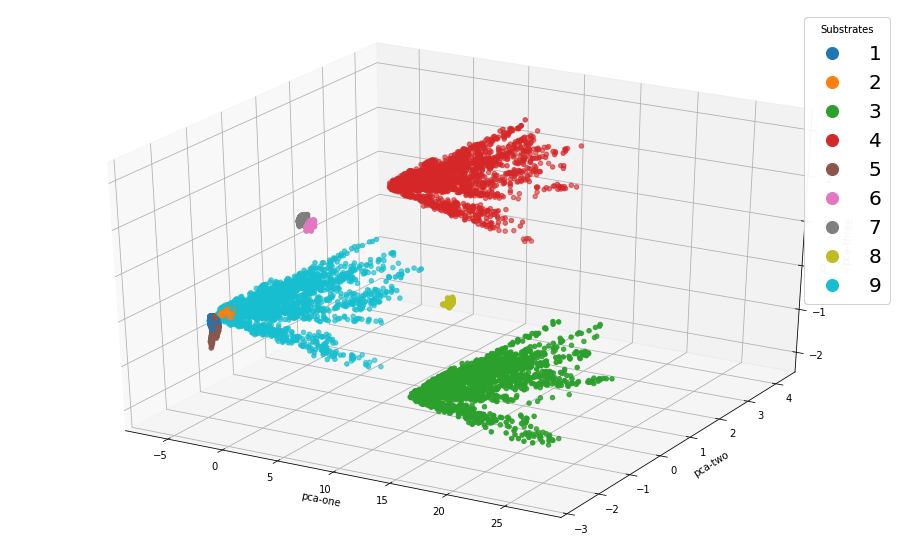
**Introduction**

The simulated data consist of 18,000 spectra of 40 Analytes taken at 1701 points in 9 substrates. There are 2,000 spectra per substrate. Each analyte has 50 spectra in each substrate. In total, there are 450 spectra of each analyte in the training dataset. This report contains three section. In the first section the raw data is visualized using PCA and t-SNE. In the second section, the data is processed. First, by removing the effect of the substrate and second by normalization. In the last section five classifiers are applied to the processed data.

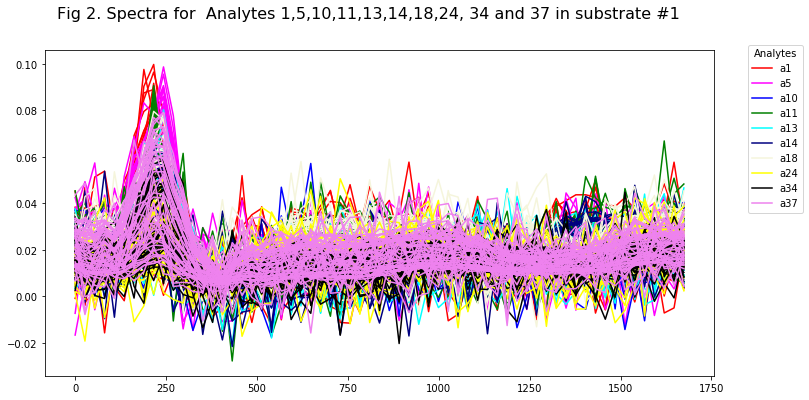
**1-Data Visualization**

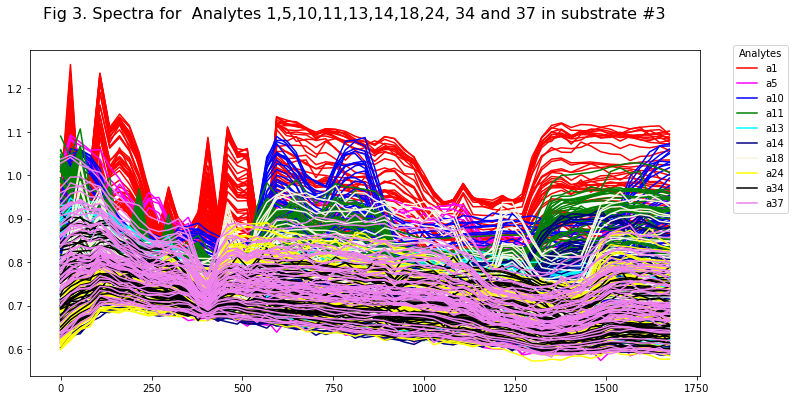
In order to have an idea of the groups formed by the 18,000 spectra we mapped the 1701 dimensional data into ether two or three dimension. In order to do that we used the classical Principal Component Analysis methodology. The first principal component represents 90.79% of the total variation and in total the first three principal components represents 99.12% of the total variation, which is very high percentage. In Figure 1, we can see the PCA visualization of the 18,000 spectra.

**Fig 1. Principal Components 3D visualization of the raw data, colored by Substrate**



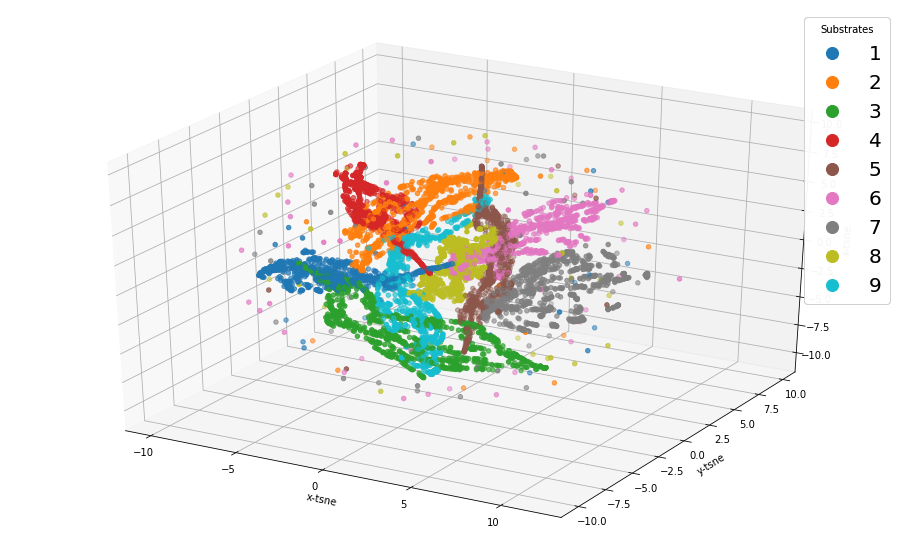
Clearly, there are three large clusters of data points corresponding to substrates 3,4, and 9. The remaining six substrates are grouped in clusters of very small volume. But, the measurements scale of the spectra in these substrates is higher than in the remaining substrates. Hence, a substrate effect must be taken into consideration. The following two figures show the spectra of some analytes in two substrates 1 (tiny size) and 3 (large size). Spectra in the substrate 3 look more spread out and this suggests an easier analyte classification. Indeed, the performance of the classifiers deteriorated at the substrates 6,7 and 8.





A nonlinear transformation called t-distributed Stochastic Neighbor Embeding (t-SNE) . The use of this transformation avoids to tendency to map a lot of points at the center of the image and reveals structure at many different scales[1]. In Fig 4, we show the t-SNE image for the 18,000 spectra. Clearly, we can notice nine clusters corresponding to the substrates. However, there are some data points around the cluster. T-SNE has not been able to map these points into a corresponding cluster due to the “curse of dimensionality” effect. To improve the visualization, the authors of t-SNE recommend to perform dimensionality reduction using PCA before to apply t-SNE. Thus, we reduce our data using 100 principal components. Figure 5, shows t-SNE for the 18,000 spectra and clearly we can identify 9 substrates. Furthermore, the outer points do not appear anymore.

**Fig 4. t- distributed Stochastic Neighbor Embedding (t-SNE) 3D visualization of the raw data, colored by substrate**



**Fig 5. t-sne visualization for the raw data after based on 100 PCs (Plotly)** A close up of a piece of paper

Description automatically generated

Since the visualization all the whole data suggest emphatically an effect due to the substrate

Where the analyte is located. In the next section, we will remove this effect before performing any classification task.

**2-Data Preprocessing**

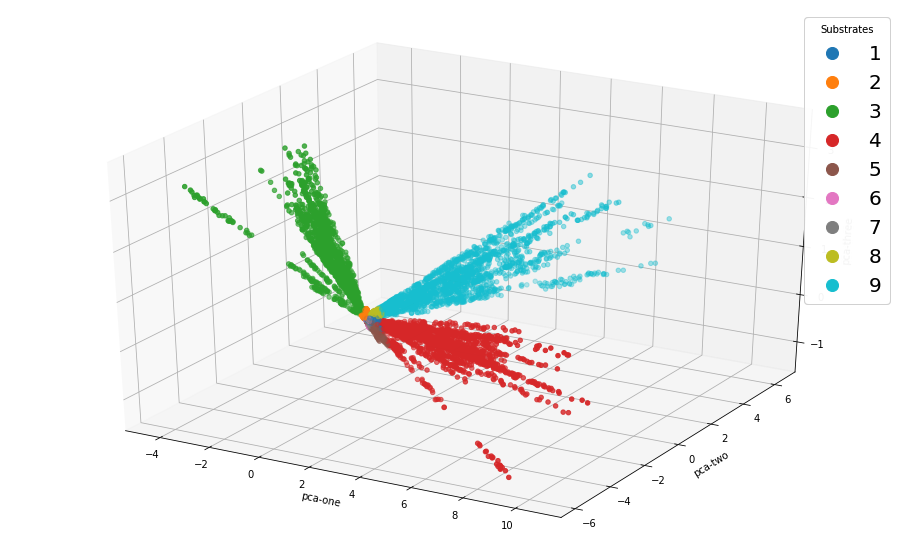
To overcome the substrate’s effect, we apply a removal of the substrate effect (centering) along with data normalization as proposed by Dr. Furstenberg. The removal equation is as follows;

**where**  stands for the raw spectra and stands for the signature of the subtrate.

**2.1 Visualization after centering**

As in the previous section, we will apply PCA and t-SNE to our centered data. First we notice that the percentage of explained variance due to the first three principal component decreases from 99.2% to 88.83%. Therefore the removal of the substrate effect is significant. Fig 6 shows the visualization of the centered data after reducing the high dimensional data to 3D using PCA.

**Fig 6. Principal Components data visualization in 3D after centering , colored by Substrate**



Still, we can only notice three large clusters. Furthermore, all the clusters intersect in one common point.

We also carry out the t-SNE visualization as is shown in Fig 7. The points falling in the perimeter around the clusters belong to the substrates 1,2,5,6,7, and 8.

**Fig 7. t-SNE 3D visualization of the centered data, colored by substrate**

A close up of a logo

Description automatically generated

**2.2 Visualization after Centering and Normalization:**

In order to have all the spectra in the same value range, we normalize them in such way that the norm of each spectra is equal to 1. We repeated the data visualization using PCA and t-SNE. But, the first 3 principal components explain only 51.5% of the total variation. The 3D PCA visualization is shown in Fig 8 an it looks very competitive with the t-SNE for the raw data. In Fig 9., we show the t-SNE visualization of the centered and normalized data in two dimensions but after reducing the dimensionality using 100 principal components. Clearly, we can see the 9 substrates.

**Fig 8. Principal Components data visualization in 3D after centering and normalization colored by Substrate**

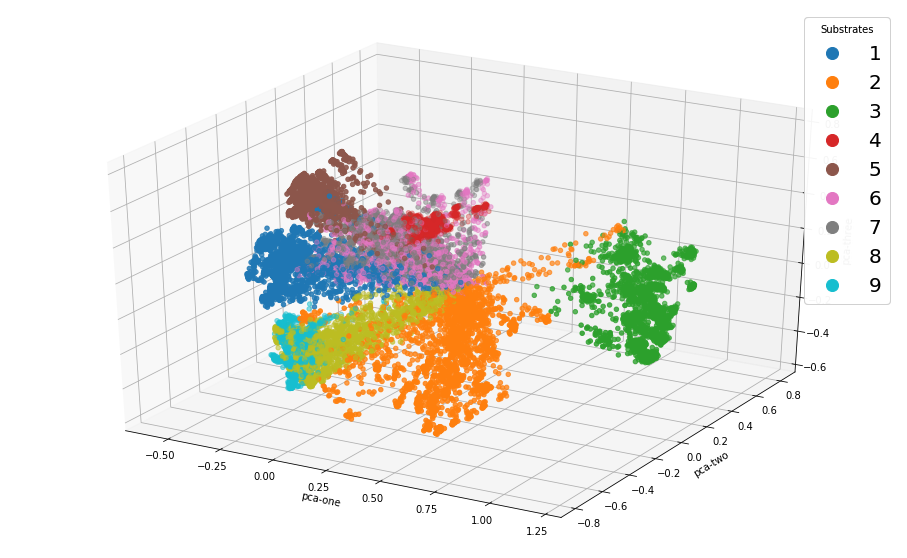
****

Fig 9. t-SNE 2D visualization of the centered and normalized data using 100 PCs

**A close up of a map

Description automatically generated**

**3-Analyte classification**

In this work, we have applied five classifiers to predict 40 analytes located in 9 substrates. A brief explanation of the classifiers used is given below. Also, we have carried out multiclass classification of all 40 analytes as well as computed metrics for binary classification considering each of the analytes as the target class and the remaining ones as the opposite class. Since the binary classification is imbalanced, we have computed the F1-score of the target class and the ROC curve. All the experiments are performed on the centered and normalized data.

**3.1. Classifiers**

1. **K-nearest neighbors**. In this case, the classification of an instance **x** is carried out as it follows:

i) Find the k instances in the training dataset that are closer in distance (Euclidean distance is the most used) to the instance **x**, k usually is and odd number: 1, 3 etc. to avoid ties.

ii) If the majority of these k instances belong to class Ci, then instance x is assigned to that class.

In case of a tie, the class is assigned randomly.

In this study, we have used k=9 for multiclass classification and k=3 for the binary classification.

The time complexity of the algorithm is O(ndk), where n=number of instances, d=dimension and k=number of neighbors.

2-**Multilayer Perceptron (MLP)**. In this work, we have use a sequential model MLP which is a linear stack of layer. This is a deep learning algorithm which is an extension of the basic neural network. The MLP’s architecture used in this work consist of:

a) An input layer containing units equal to the number of predictors.

b) A Dropout layer, where a fraction of inputs is set up to zero in an effort to reduce overfitting[2]. A .30 dropout was applied to the input layer meaning that in each epoch a 30% of features was discarded at random from the parameter estimation.

c) Two Dense connected layers. In the multiclass classification task, the first layer has 300 neurons and the second layer has100 neurons.

d) Epochs: is the number of times that the model is exposed to the training dataset. In this work we have used 100 epochs.

e) Batch Size: is the number of training instances shown to the model before a weight update is performed. A batch size of size 150 was used in this work.

We have done a lot of tuning regarding the parameters of our networks. There is another parameter called the learning rate, but we used the one that has been set by default.

The time complexity for an MLP approximate solution is O(epochs \* n \* d \* #neurons).

3. **Non-linear Support vector Machine (SVM).** The classification is based on the instances closest to the boundary (hyperplane) between classes. These instances are called support vectors. The goal is to maximize the Margin, which is the distance between support vectors belonging to opposite classes .

The general Idea is the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable, using a function called the Kernel. In our study, we have used a Gaussian kernel ( also called Radial basis Function) [3].

SVMs are inherently two-class classifiers. The traditional way to do multiclass classification with SVMs is to build C one-versus-rest classifiers (commonly referred to as ``one-versus-rest' or OVR classification), and to choose the class which classifies the test datum with greatest margin.

The time complexity of the SVM is given by O(n2).

4**- Logistic Regression**: For binary classification in class C1=1 and class C2=0, It is assumed that the ratio of distribution of the predictors in each of the classes holds this condition,



where **x** is a p-dimensional vector of predictor variables, α is a constant and β is a vector of p parameters. The estimation of the parameters is done iteratively. In here, we have used the Newton-conjugate gradient method with 10,000 iterations.

Let p=P(Y=1/x) be the posterior probability that an instance with observed measurements x belong to class Y=1. The classification is based on the estimated value of p.



When there are K classes then the odds ratios are taken with respect to one class, say the last class. Thus,

Log[P(Y=j/x)/P(Y=K/x)]=αj+β’j**x** para j=1,2,….K-1

Hence,



for j=1,2,…K-1, and,



In the area of neural networks these functions are called softmax. The instance x is assigned to class j\* such that j\*=argmax(pj). The time complexity of the Logistic regression is given by O(d\*c), where c is the number of classes.

**5-Convolutional Neural networks (CNN).** There are plenty of CNN architectures: LeNet, AlexNet, GoogleNet[4]. Here, we have used a sequential model for CNN. The first layer considered was a one-dimensional convolutional layer,  **Conv1D.** 1D convolutional layers are efficient for analyzing time series data as well spectral data. In this work, since each row of the data frame is converted in a vector of shape (1701,1), this is going to be the input shape parameter of the Conv1D layer. Then, we added Dense, MaxPooling1D, and Flatten layers into the model. The output layer contains the number of output classes, 40, and the 'softmax' activation function. As in the MLP algorithm, we used 100 epochs. The batch size used was 64.

**3.2 Results for Multiclass classification**

In our 18,000 instances, 1701 features and 40 classes. Training set 80%, Test set: 20%.

**Table 1. Metrics for Multiclass classification**

|  |  |  |  |
| --- | --- | --- | --- |
| **Classifier** | **F1-score** | **Accuracy** | **Running time(secs)** |
| Knn (k=9)\* | .9822 | .9822 | 3.7 |
| CNN | .9657 | .9655 | 454.4 |
| MLP | .9503 | .9502 | 331.6 |
| SVM | .9370 | .9277 | 207 |
| Logistic Regression | .9644 | .9636 | **8.4** |

\*k=9 neighbors was chosen based on 25 runs (train/test), k was varied from 1 up to 40.

From Table 1, k-nn seems to have the best metrics values. In second place is CNN. A validation set of 20 percent was used for MLP and CNN. This fact decreases the training size from 18,000 to 11,520 instances, since 3,600 are already in the test dataset and from the remaining 14,400 instances 2880 are assigned to the validation set. This can be the reason because k-nn outperforms CNN for a little bit.

The fastest algorithm was Logistic Regression, whereas MLP was the slowest one. The centering and normalization of the data improves the performance of MLP and logistic regression classifiers.

Next, in order to speed up the computation, we applied PCA with either 30 or 50 principal components.

**Table 2. Metrics for Multiclass classification after PCA**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Classifier** | **F1-score** | | **Accuracy** | | **Running time(secs)** | |
| 30 PCs | 50 PCS | 30 Pcs | 50 PCs | 30 PCs | 50 PCs |
| K-nn | 0.9484 | 0.9405 | 0.9480 | 0.9400 | 0.36 | 1.39 |
| CNN | 0.9588 | 0.9562 | 0.9588 | 0.9572 | 11.43 | 15.14 |
| MLP\* | 0.9492 | 0.9494 | 0.9492 | 0.9491 | 403.8 | 198.9 |
| SVM | 0.9557 | 0.9343 | 0.9552 | 0.9322 | 6.52 | 8.86 |
| Logistic Regression | 0.9530 | 0.9654 | 0.9525 | 0.9647 | 2.50 | 2.10 |

\* MLP with two layers each of them having 30 neurons

All the classifiers had similar performance in both accuracy and F1-score. Furthermore, the running time decreased in all of them, but MLP which still has a very high running time.

**3.2 Metrics for binary classification**

Forty binary classification were performed. The target class contains 450 instances corresponding to a given analyte, the opposite class contains 17450 instances. Since, this is an imbalanced problem, only F1-score and ROC curves are reported. The F1-score is simply the harmonic mean of precision (PRE) and recall (REC)

F1 = 2 \* (PRE \* REC) / (PRE + REC)

The F1-score metric tries to find an equal balance between precision and recall, which is extremely useful in most scenarios when we are working with imbalanced datasets. If we write the two metrics PRE and REC in terms of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), we get:

* PRE = TP / (TP + FP)
* REC = TP / (TP + FN)

For instance, suppose that we want to classify the j-th analyte (j=1,…40). Then, the precision gives us an idea of the proportion of how many actual j-th analytes (TP) we correctly classified among all the analytes we classified as the j-th analyte (TP + FP). In contrast, the recall tells us about the proportion of how many of the actual j-th analytes (TP) we “retrieved” or “recalled” (TP + FN). Table 3 shows the F1-score for the 40 analytes using five classifiers.

**Table 3. F1-score for binary classification after PCA**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Analyte | Knn | SVM | MLP | CNN | Logistic Regression | Average  F1-score |
| 1 | 1.000 | 0.957 | 0.9756 | 0.9988 | 0.982 | 0.9826 |
| 2 | 0.991 | 0.935 | 0.9661 | 0.9988 | 0.971 | 0.9723 |
| 3 | 0.995 | 0.897 | 0.9281 | 0.9966 | 0.903 | 0.9439 |
| 4 | 0.983 | 0.854 | 0.9176 | 0.9986 | 0.885 | 0.9276 |
| 5 | 0.995 | 0.927 | 0.9801 | 0.9991 | 0.948 | 0.9698 |
| 6 | 0.981 | 0.940 | 0.9808 | 0.9991 | 0.975 | 0.9651 |
| 7 | 0.988 | 0.938 | 0.9634 | 0.9994 | 0.976 | 0.9729 |
| 8 | 0.989 | 0.929 | 0.9714 | 0.9977 | 0.966 | 0.9706 |
| 9 | 0.994 | 0.903 | 0.9261 | 0.9991 | 0.925 | 0.9494 |
| 10 | 0.976 | 0.920 | 0.9811 | 0.9988 | 0.933 | 0.9617 |
| 11 | 1.000 | 0.900 | 0.9846 | 0.9969 | 0.944 | 0.9651 |
| 12 | 0.978 | 0.881 | 0.9239 | 0.9936 | 0.700 | 0.8953 |
| 13 | 0.951 | 0.770 | 0.8947 | 0.9986 | 0.567 | 0.8362 |
| 14 | 0.979 | 0.855 | 0.9444 | 0.9990 | 0.783 | 0.9120 |
| 15 | 0.980 | 0.872 | 0.9513 | 0.9959 | 0.893 | 0.9384 |
| 16 | 0.944 | 0.837 | 0.8666 | 0.9980 | 0.840 | 0.8971 |
| 17 | 0.984 | 0.869 | 0.9204 | 0.9971 | 0.892 | 0.9325 |
| 18 | 0.971 | 0.861 | 0.9512 | 0.9980 | 0.735 | 0.9032 |
| 19 | 0.991 | 0.878 | 0.9565 | 0.9977 | 0.961 | 0.9568 |
| 20 | 0.954 | 0.916 | 0.9569 | 0.9988 | 0.753 | 0.9157 |
| 21 | 0.994 | 0.976 | 0.9940 | 0.9991 | 0.933 | 0.9792 |
| 22 | 0.995 | 0.922 | 0.9753 | 0.9983 | 0.957 | 0.9695 |
| 23 | 0.989 | 0.910 | 0.9600 | 0.9941 | 0.949 | 0.9604 |
| 24 | 0.980 | 0.881 | 0.9304 | 0.9974 | 0.729 | 0.9035 |
| 25 | 0.988 | 0.892 | 0.9764 | 0.9986 | 0.748 | 0.9206 |
| 26 | 1.000 | 0.867 | 0.9419 | 0.9963 | 0.906 | 0.9422 |
| 27 | 0.982 | 0.897 | 0.9202 | 0.9986 | 0.828 | 0.9251 |
| 28 | 0.990 | 0.921 | 0.9591 | 0.9991 | 0.965 | 0.9668 |
| 29 | 0.996 | 0.915 | 0.9753 | 0.9985 | 0.969 | 0.9707 |
| 30 | 0.994 | 0.916 | 0.9574 | 0.9974 | 0.866 | 0.9461 |
| 31 | 0.967 | 0.832 | 0.9111 | 0.9971 | 0.848 | 0.9110 |
| 32 | 0.989 | 0.828 | 0.9195 | 0.9980 | 0.796 | 0.9061 |
| 33 | 0.987 | 0.841 | 0.9220 | 0.9954 | 0.850 | 0.9190 |
| 34 | 0.960 | 0.795 | 0.9122 | 0.9980 | 0.785 | 0.8900 |
| 35 | 0.988 | 0.845 | 0.9554 | 0.9968 | 0.962 | 0.9494 |
| 36 | 0.988 | 0.892 | 0.9554 | 0.9971 | 0.696 | 0.9057 |
| 37 | 0.982 | 0.779 | 0.9161 | 0.9977 | 0.750 | 0.8849 |
| 38 | 0.983 | 0.886 | 0.9461 | 0.9985 | 0.965 | 0.9557 |
| 39 | 0.986 | 0.939 | 0.9710 | 0.9983 | 0.971 | 0.9730 |
| 40 | 0.985 | 0.900 | 0.9411 | 1.0000 | 0.948 | 0.9548 |
| Average | 0.9836 | 0.8868 | 0.9470 | 0.9978 | 0.8738 |  |

From Table 3, clearly CNN outperforms all the other classifiers. Only k-nn beats CNN in three analytes.

On average, analytes #12, #13, #16, #34, and #37 are the hardest to be classified.

**Table 4. Running Time (in seconds) to perform all 40 analytes classification**

|  |  |  |
| --- | --- | --- |
| **Classifier** | **Without PCA** | **Using 50 PCs** |
| SVM | 2,021 | 9.87\* |
| k-nn | 8,139 | 13.9\* |
| MLP | 60,000 | 56.400 |
| CNN | 81,000 | 4,800 |
| Logistic Regression | 228 | 6.71 |

\*30 PCs where used for these classifiers

An ROC curve (or receiver operating characteristic curve) is a plot that summarizes the performance of a binary classification model on the positive class. The x-axis indicates the False Positive Rate and the y-axis indicates the True Positive Rate

**True Positive Rate** = TP / (TP + FN). Same as Recall o Sensitivity

**False Positive Rate** = FP/ (FP + TN)

By evaluating the true positive and false positives for different threshold values, the ROC curve is obtained.

The ideal situation occurs when the fraction of correct positive class predictions is 1 (top of the plot) and the fraction of incorrect negative class predictions is 0 (left of the plot). The area under the curve (auc) can be calculated to give a single score for a classifier model across all threshold values. The value of auc is equal to 1 for a perfect classifier.

Figures 10,11, and 12 show the ROC curves for the k-nn, SVM and Logistic regression classifiers.

**Fig 10. ROC curve for the k-nn classifier**

A screenshot of a cell phone

Description automatically generated

Fig 11. **ROC curve for the Logistic Regression classifier**

A screenshot of a cell phone

Description automatically generated

Fig12 **ROC curve for the non-linear SVM classifier**

.A screenshot of a cell phone

Description automatically generated

Consistently, analytes #13, #34 and #37 are hard to be classified. However, these are not that ones with the highest noise according to theirs average mass loadings. Analytes #28, #26 and #29 are the ones with the lowest average mass loadings.

The metrics or the imbalanced classific

**Acknowledgment**

I would like to thank Dr. Furstenberg and Dr. Kendziora researchers from the Naval Research

Laboratory for their constant support during this research work.

**References:**

1. Maaten, L.V. and an Hinton, G. (2008). **Visualizing data using et-SNE** Journal of Machine Learning Research, Vol 9 (November), pp. 2579-2605.
2. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting.Journal of Machine Learning Research. Vol15, 56.,pp1929-1958.
3. Cortes, C. and Vapnik, V N. (1995). Support Vector Networks. [*Machine Learning*](https://en.wikipedia.org/wiki/Machine_Learning_(journal)). **20** (3): 273–297.
4. Khan, A., Sohail, A., Zahoora, U. *et al.* A survey of the recent architectures of deep convolutional neural networks. *Artif Intell Rev* (2020). https://doi.org/10.1007/s10462-020-09825-6