

---

# EPILEPSY DETECTION

PREPARED BY

BEN OTHMEN WALID

DAGHBOUJI MED AMINE

BRAD STEVE

KONYALI MED AMINE

BOUSLEH MALEK

STITI BACEM

---

# Table of Contents

<b>Table of Contents .....</b>	<b>2</b>
<b>Introduction: .....</b>	<b>4</b>
<b>1. Dataset: .....</b>	<b>5</b>
1.1. Internal Data: .....	5
1.1.1. Isip Dataset: .....	5
1.1.2. Bonn Dataset: .....	5
1.2. External data: .....	6
<b>2. Pre-processing: .....</b>	<b>7</b>
2.1. Isip Dataset: .....	7
2.1.1. Feature extraction: .....	8
2.1.2. Bonn Dataset: .....	10
2.2. External data: .....	12
<b>3. Machine Learning Classifiers: .....</b>	<b>13</b>
3.1. K Nearest Neighbors: .....	13
3.2. LinearSVM: .....	14
3.3. Decision tree classifier: .....	15
3.4. Random Forest: .....	15
3.5. Deep Neural Networks: .....	16
3.5.1. Artificial Neural Network: .....	16
3.5.2. LSTM: .....	17
3.5.3. Sequential: .....	18
<b>4. Evaluation: .....</b>	<b>19</b>
4.1. Tools: .....	19
4.1.1. Bonn models evaluation: .....	19
4.1.1.1. Binary models(Epileptic vs Healthy): .....	19
4.1.1.2. Multiclass classification models (Seizure vs Tumor area vs Healthy area): .....	20

4.1.2. Isip models evaluation: .....	21
4.1.2.1. Knn Model Evaluation:.....	21
4.1.3. External data models evaluation: .....	22
<b>5. Limitations and Future Work: .....</b>	<b>24</b>
<b>6. Conclusion: .....</b>	<b>25</b>
<b>7. References: .....</b>	<b>26</b>

# Introduction:

Epilepsy is a chronic neurological disorder characterized by frequent seizures, which severely impacts the quality of life of epileptic people, sometimes it is accompanied by loss of consciousness. There can be many causes of epilepsy and sometimes it is not possible to identify them. In the domain of epilepsy, seizure is referred to as an epileptic seizure and brain is the source. During an epileptic seizure normal functioning of the brain is disturbed for that certain time period, causing disruption on signaling mechanism between brain and other parts of the body. These seizures can put epilepsy patients at higher risk for injuries including fractures, falls, burns and submersion injuries, which are very common in children. These injuries happen because seizure can happen anytime and anywhere without prior warning and the sufferer would continue his or her activity with an unconscious mind.

The most widely accepted and used tool by epileptologists to identify and diagnose epilepsy is the ElectroEncephaloGram (EEG).

Epilepsy detection on EEG signals is a long process, which is done manually by epileptologists ,our main goal is to automate this process.

# Dataset:

## 1.1. Internal Data:

### 1.1.1. Isip Dataset:

This is a subset of TUEG that contains 100 subjects epilepsy and 100 subjects without epilepsy, as determined by a certified neurologist. The data was developed in collaboration with a number of partners including NIH.

### 1.1.2. Bonn Dataset:

The dataset for this study is generated from the Epileptic Seizure Recognition Data Set under UCI Machine Learning repository [4]. The dataset contains 500 patients' 4097 electroencephalograms (EEG) readings over 23.5 seconds. The 4097 data points are reshaped into 23 chunks with each chunk including 178 voltage signals corresponding to the brain activity within one second. Consequently, this multivariate time series dataset has 11500 subjects with each subject having a brain activity label regarding 178 features. 5 kinds of brain activities are seen in the dataset. We randomly select an instance under each category and visualize it in Fig. The detailed description of each activity can be found below. We also denote each class an abbreviation name:

**Seizure:** Seizure activity is recorded.

**Tumor Area:** Subject is diagnosed with tumor, and the EEG is collected from these epileptic brain areas.

**Health Area:** Subject is diagnosed with tumor however the EEG is collected from non-epileptic brain area.

**Eyes Closed:** Collected from the healthy subject when the eyes are keeping close

**Eyes Open:** Collected from the healthy subject when the eyes are keeping open.

## 1.2. External data:

The mock dataset was established to assess the classification algorithm processing. It is composed of 3 datasets each for a different patient.

xTemperature	xSound	xAcceleromet	xGyroscope	xEDA	xHeartbeat	Y
37,2	40	36	10	0,2	77	0
39	102	110	14	1,3	115	1
36,5	46	63	6	0,6	68	0
41	30	102	15	2	123	1
39,1	50	67	10	0,9	100	0
39,6	32	132	16	3	129	1
40	116	64	11	0,4	100	0
38,8	32	126	12	1,5	130	1
36,8	12	99	64	0,6	78	0

The datasets were prepared in steps.

First step was to identify biosensors; therefore, a qualitative research was conducted with epileptic patients, neurologists and pediatricians in Institute of Neurology and Association Ettafaouel.

The selected biosensors were Temperature, Heartbeat, Accelerometer, Gyroscope, Electro-dermal Activity and environmental sound.

The second step was to identify the thresholds of every biosensor in condition of generalized seizures occurring to epileptic patients. For the simplicity of the research, predefined thresholds were fixed according to every biosensor.

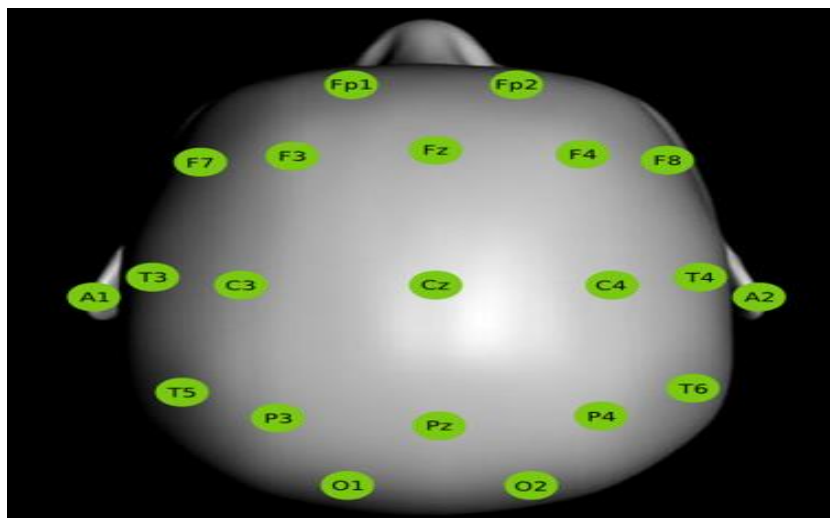
Our external data is based on the combination of the values that we got from using many types of sensors like temperature, sound of the atmosphere, accelerometer, gyroscope, EDA and heartbeat.

# Pre-processing:

## 2.1. Isip Dataset:

Bad channels remove:

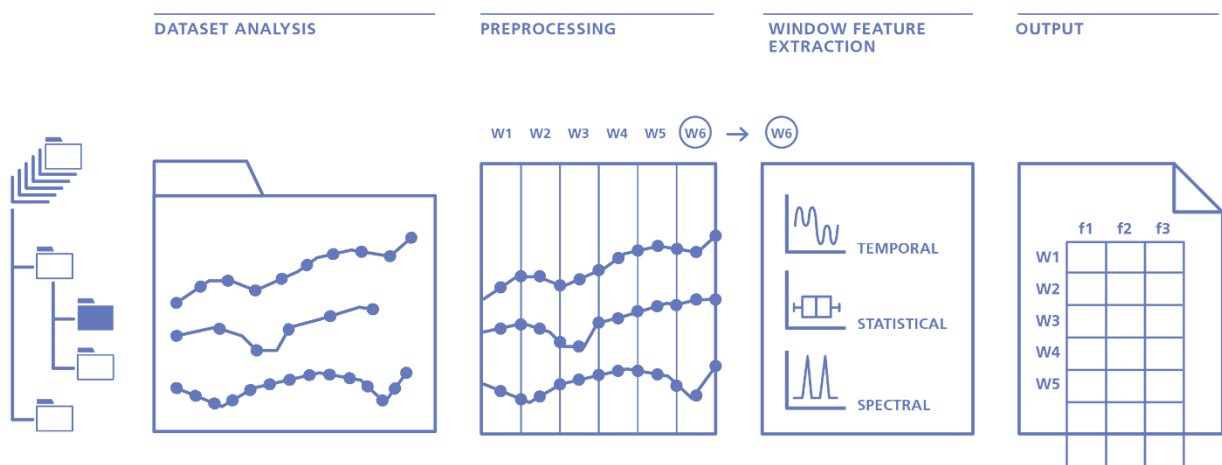
The TUH EEG Corpus contains data collected with a minimum of 24 channels (21 signals plus three annotation channels) and 36 channels. Our baseline experiments focus only on the first 21 channels.



### 2.1.1. Feature extraction:

To extract features, we splitted the signals into several windows and we extracted more than 60 features from each window using TSFEL.

Time Series Feature Extraction Library (TSFEL for short) is a Python package for feature extraction on time series data. It provides exploratory feature extraction tasks on time series without requiring significant programming effort. TSFEL automatically extracts over 60 different features on the statistical, temporal and spectral domains.



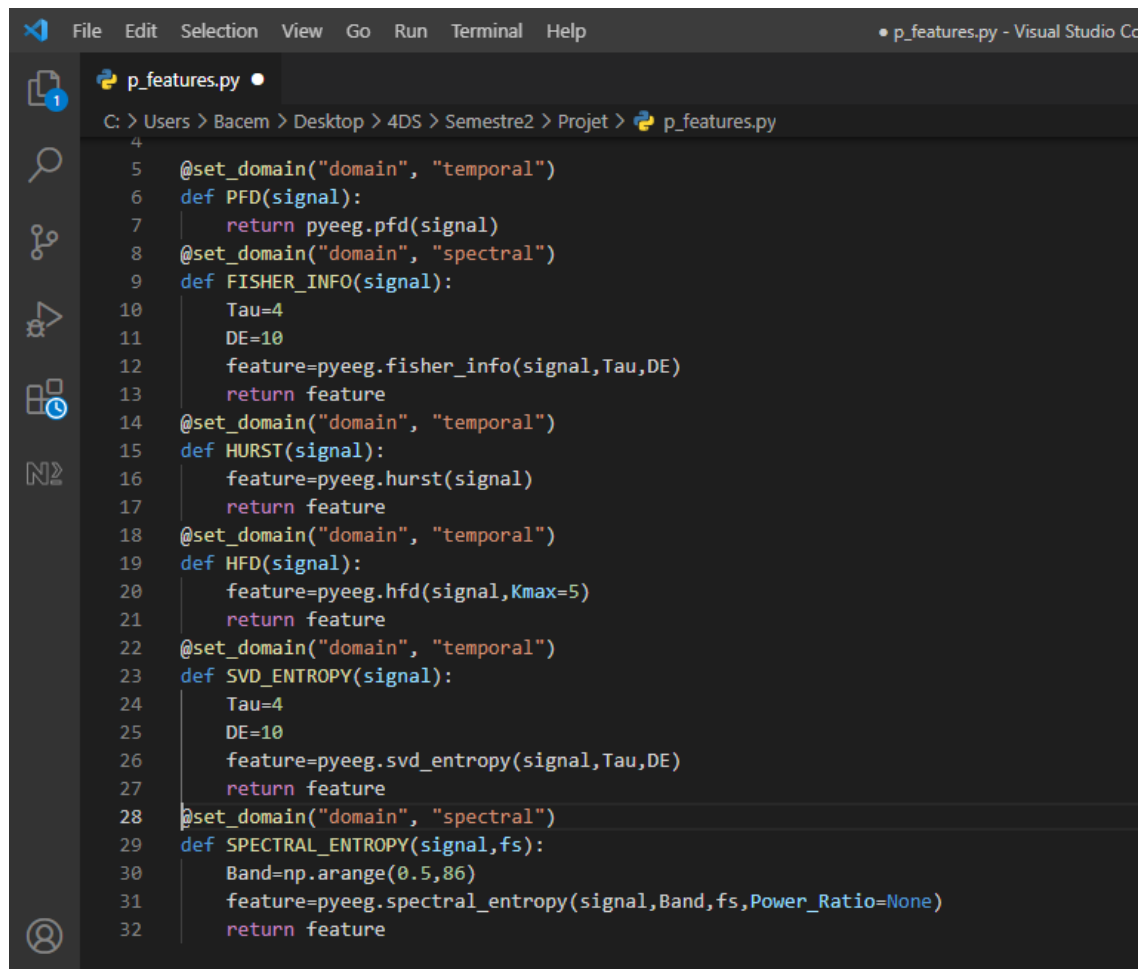
We also added some personalized features to those that are already available in the library using another library called

This is a list of the features that are available in PyEEG:

Power Spectral Intensity (PSI) and Relative Intensity Ratio (RIR)	<code>bin_power()</code>	Two 1-D vectors
Petrosian Fractal Dimension (PFD)	<code>pdf()</code>	A scalar
Higuchi Fractal Dimension (HFD)	<code>hfd()</code>	A scalar
Hjorth mobility and complexity	<code>hjorth()</code>	Two scalars
Spectral Entropy (Shannon's entropy of RIRs)	<code>spectral_entropy()</code>	A scalar
SVD Entropy	<code>svd_entropy()</code>	A scalar
Fisher Information	<code>fisher_info()</code>	A scalar
Approximate Entropy (ApEn)	<code>ap_entropy()</code>	A scalar
Detrended Fluctuation Analysis (DFA)	<code>dfa()</code>	A scalar
Hurst Exponent (Hurst)	<code>hurst()</code>	A scalar



And this is how we implemented them into TSFEL:



```
File Edit Selection View Go Run Terminal Help • p_features.py - Visual Studio Co
p_features.py
C: > Users > Bacem > Desktop > 4DS > Semestre2 > Projet > p_features.py
4
5 @set_domain("domain", "temporal")
6 def PFD(signal):
7     return pyeeg.pfd(signal)
8 @set_domain("domain", "spectral")
9 def FISHER_INFO(signal):
10     Tau=4
11     DE=10
12     feature=pyeeg.fisher_info(signal,Tau,DE)
13     return feature
14 @set_domain("domain", "temporal")
15 def HURST(signal):
16     feature=pyeeg.hurst(signal)
17     return feature
18 @set_domain("domain", "temporal")
19 def HFD(signal):
20     feature=pyeeg.hfd(signal,Kmax=5)
21     return feature
22 @set_domain("domain", "temporal")
23 def SVD_ENTROPY(signal):
24     Tau=4
25     DE=10
26     feature=pyeeg.svd_entropy(signal,Tau,DE)
27     return feature
28 @set_domain("domain", "spectral")
29 def SPECTRAL_ENTROPY(signal,fs):
30     Band=np.arange(0.5,86)
31     feature=pyeeg.spectral_entropy(signal,Band,fs,Power_Ratio=None)
32     return feature
```

## 2.1.2. Bonn Dataset:

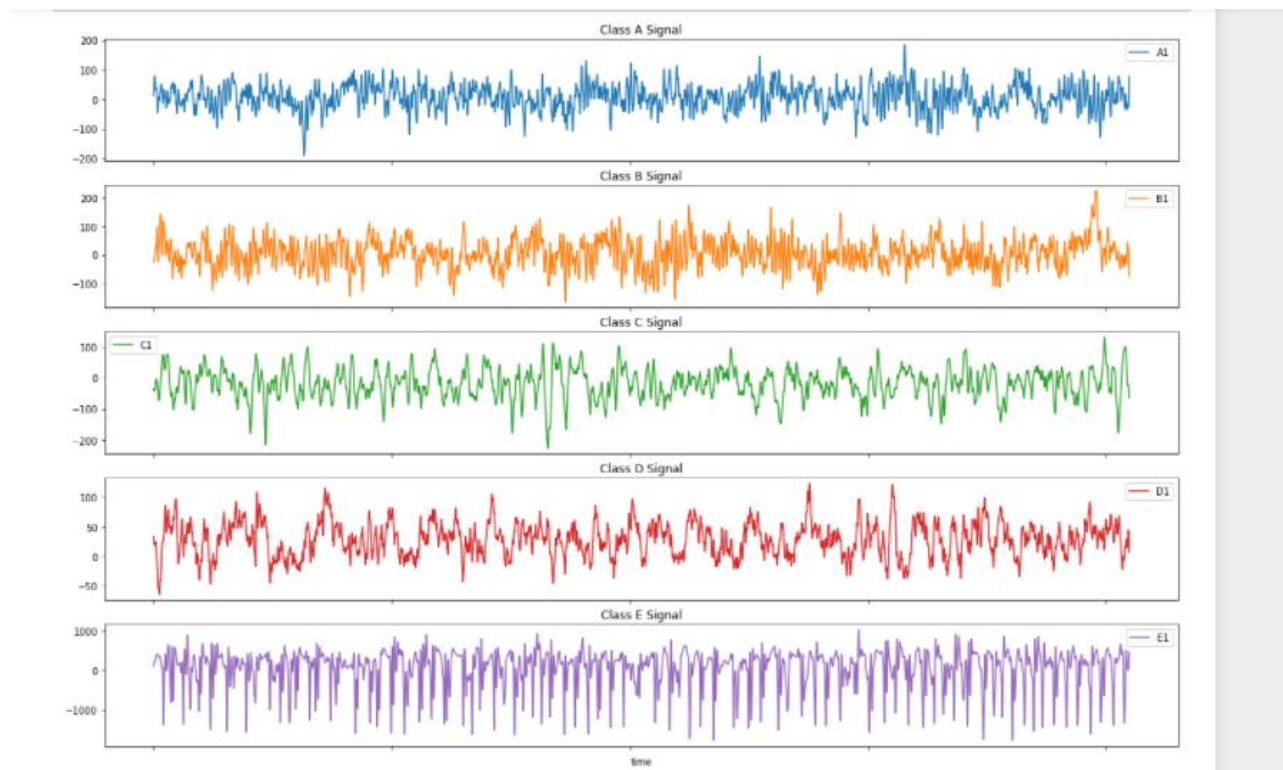
The bonn's university dataset consists of 5 classes saved in 5 different folders, each with 100 files, with each file representing a single subject so we assembled all those files into a single dataframe. and as we know the sampling frequency, we setted a time axis

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	...	E91	E92	E93	E94	E95	E96	E97	E98	E99	E100
time																					
00:00:00	12	-56	-37	-31	14	-87	-2	-31	8	-41	...	-129	-26	308	-155	-113	-40	187	-438	-476	23
00:00:00.005760	22	-50	-22	-43	26	-89	20	-16	17	-38	...	-309	1	367	-283	-185	-58	44	-561	-518	144
00:00:00.011520	35	-64	-17	-39	32	-73	42	10	29	-31	...	-432	29	413	-456	-269	-75	-147	-622	-521	228
00:00:00.017280	45	-91	-24	-39	25	-69	48	28	46	-25	...	-412	41	429	-541	-328	-88	-368	-581	-362	260
00:00:00.023040	69	-135	-31	-9	16	-51	27	31	50	-32	...	-278	33	400	-474	-312	-89	-550	-460	-68	255
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
00:00:23.570070	-25	-172	-3	11	-55	32	-38	0	-48	38	...	-44	-205	-1547	93	141	-3	748	40	224	-272
00:00:23.575830	-28	-180	7	12	-58	37	-23	3	-50	22	...	95	-209	-1023	26	171	-10	763	-47	299	-272
00:00:23.581590	-11	-173	3	-6	-32	18	-18	1	-40	1	...	12	-207	-557	-27	148	-13	703	-118	246	-155
00:00:23.587350	8	-162	4	10	-6	7	-6	1	-36	-13	...	-254	-210	-305	-85	111	-16	446	-163	556	6
00:00:23.593110	77	-82	82	33	-17	7	-37	0	7	-1	...	-57	-107	-859	-449	-235	-151	-537	-56	276	-221

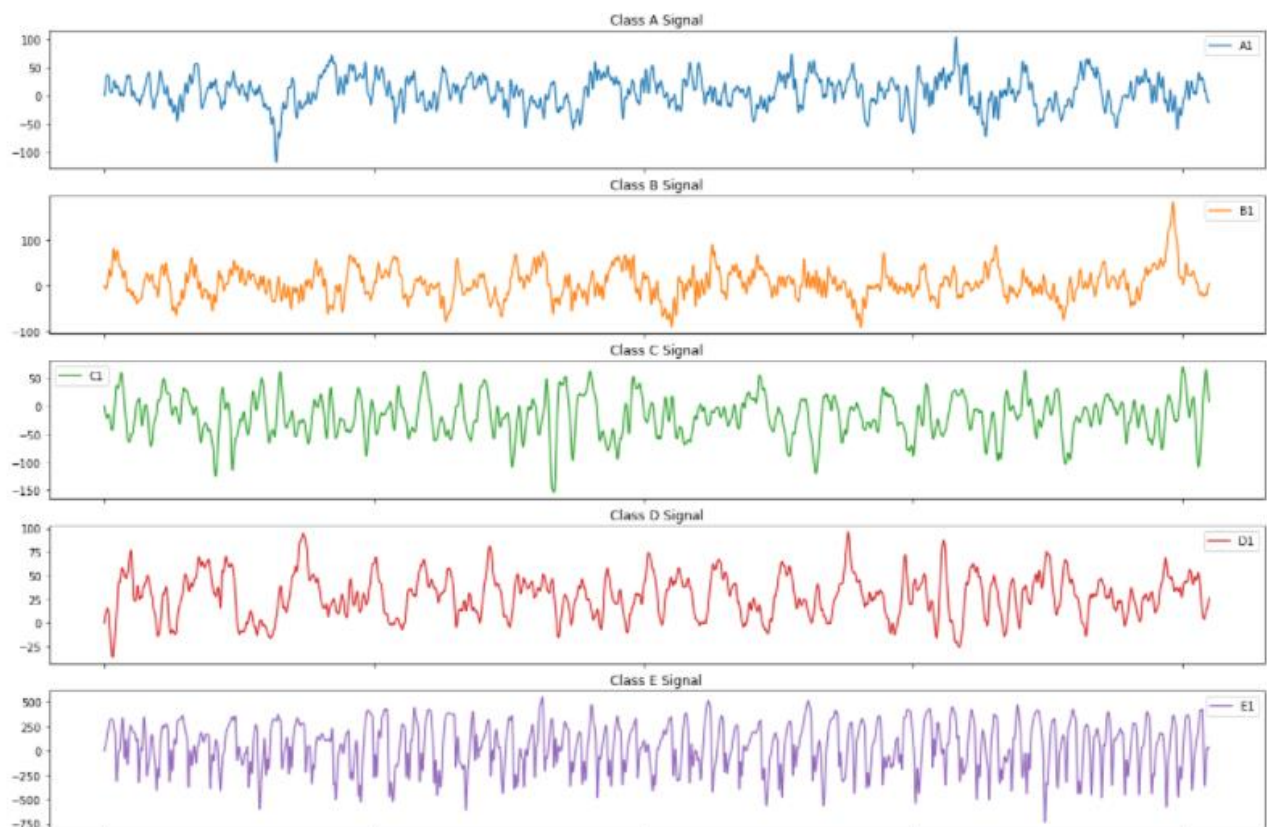
4097 rows × 500 columns

Finally we filtered all signals to reduce and smooth out high-frequency noise.

This figure shows how signals looks like before filtering:



and this one shows filtered signals:



## 2.2. External data:

During the preparation phase we found out that there are many missing and false values and sometimes unlabeled rows.

And in order to fix these problems, we filled the missing values with the mean value of the variables. For the false values, we replaced them with new logical values and we omitted the rows that seem to influence negatively our data modeling.

Also, after studying the correlation between the variables and the target we found, as we can see in the figures below, that the two variables “Sound” and “Temperature” do not have direct impact on seizures so we can delete them.

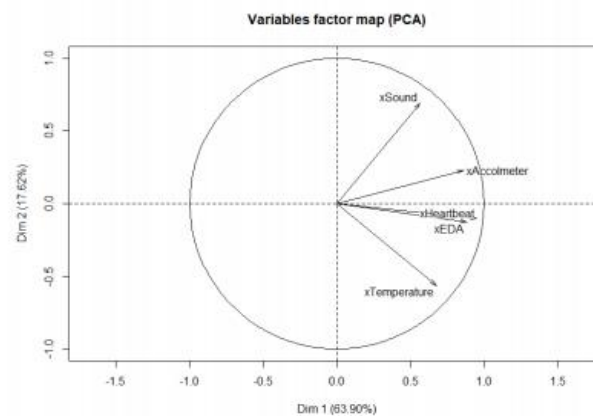


Fig. 3.1. Multivariate data analysis using PCA on data set #1

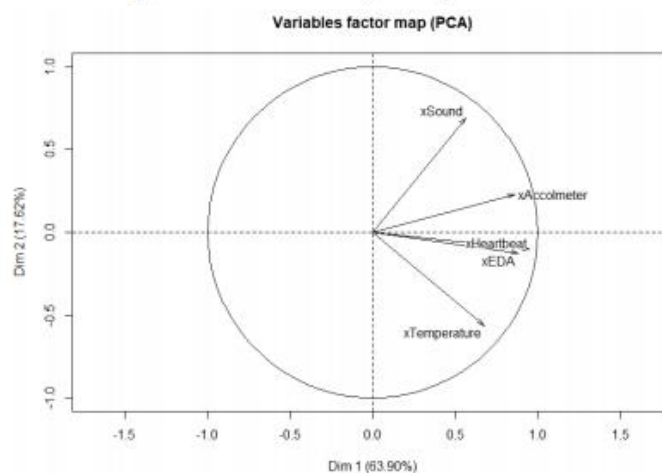


Fig. 3.3. Multivariate data analysis using PCA on data set #3

# Machine Learning Classifiers:

We employed 4 machine learning classification methods to build the predictive model. Following classifiers were experimented: linear classifiers: Support Vector Machine with linear boundary (denoted as LinearSVM) and Decision tree classifier, Random Forest (RF), and K Nearest Neighbors (KNN).

## 3.1. K Nearest Neighbors:

KNN is a very intuitive model. The sample is classified based on the labels of its K nearest neighbors. Another advantage of the model is time efficiency.

There is almost no training time due to its principle. Euclidean distance was used to determine the neighbors.

We divided our data into 80% learning data and 20% test data and we used oversampling to solve the data imbalance problem

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X_Sampled,Y_Sampled,test_size=0.2,random_state=1)
X_train.shape, X_test.shape,y_train ,y_test
```

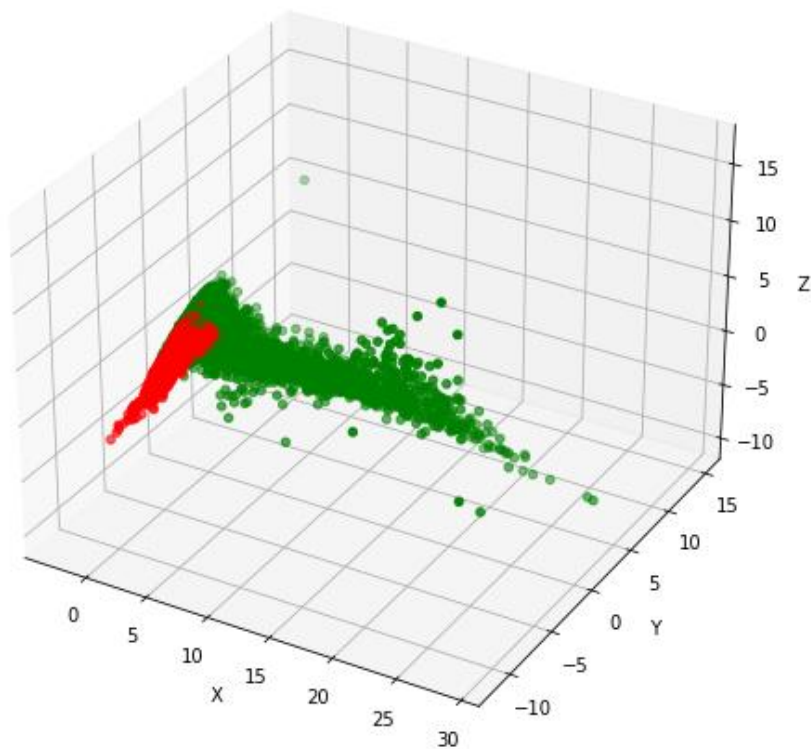
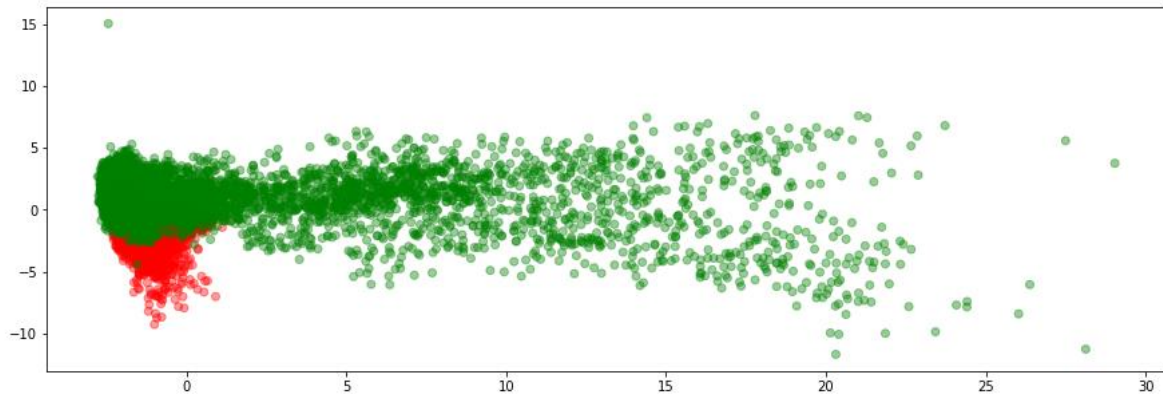
```
▶ X= ds.drop(['Tag'],axis=1)
  y= ds['Tag']
  clusters = RandomOverSampler(sampling_strategy='minority',random_state =1)
  clusters.fit(X,y)
  X_Sampled,Y_Sampled = clusters.fit_resample(X,y)
```

```
[ ] from sklearn.neighbors import KNeighborsClassifier
    knn = KNeighborsClassifier(8,p=1,leaf_size=1 )
    knn_model = knn.fit(X_train, y_train)
    y_pred_knn = knn_model.predict(X_test)
```

### 3.2. LinearSVM:

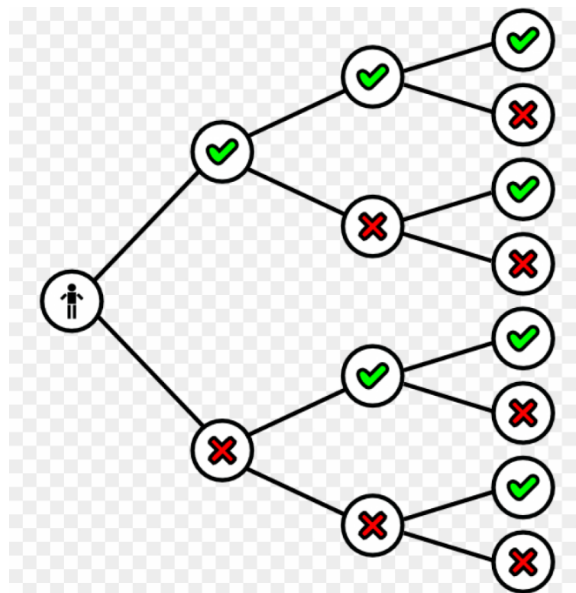
An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. L2-regularization, One-versus-All were also applied. As there is non- probabilistic, we converted the distance to decision boundary to form a probabilistic output.

We applied a 2D and 3D PCA to visualize the distribution of data.



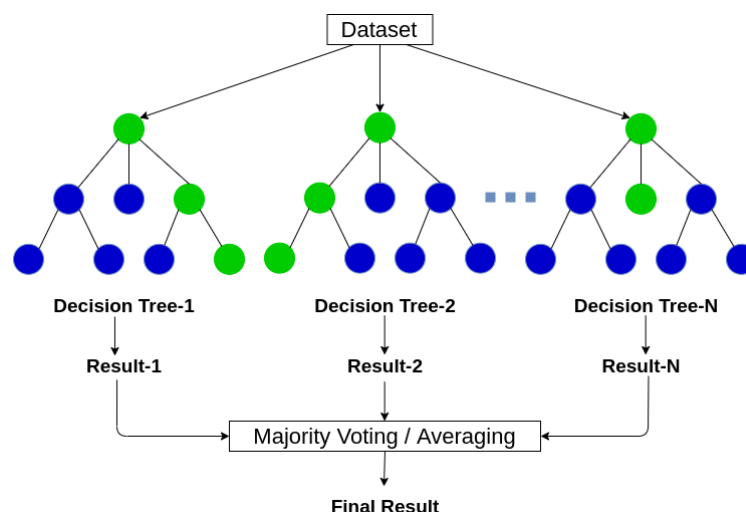
### 3.3. Decision tree classifier:

A decision tree is a managerial tool that presents all the decision alternatives and outcomes in a flowchart type of diagram, like a tree with branches and leaves. Each branch of the tree represents a decision option, its cost and the probability that it is likely to occur. The leaves at the end of the branches show the possible payoffs or outcomes. A decision tree illustrates graphically all the possible alternatives, probabilities and outcomes and identifies the benefits of using decision analysis.



### 3.4. Random Forest:

A random forest consists of multiple decision trees, which bootstrap from subset randomly sampled entire dataset, to reduce the probability of over-fitting. We also limited the feature amounts under the square root of the complete set. Given that random forest classifier has many parameters to tune, we only tune two mainly factors: amount of estimators and max depth of each tree.



## 3.5. Deep Neural Networks:

### 3.5.1. Artificial Neural Network:

Artificial neural networks (ANNs) are applied in a variety of scientific fields, such as medical diagnosis, speech and pattern recognition, etc. ANN is a computing scheme partly representing the biological neural networks existing in human or animal brains, expressed by connected nodes (artificial neurons) organized properly in layers. All artificial neurons are connected and able to transmit signals, usually real numbers, through their connections (synapses), resulting in an output calculated suitably by a nonlinear function according to the initial inputs based on specific weights assigned to all neurons.

---

Model: "sequential"

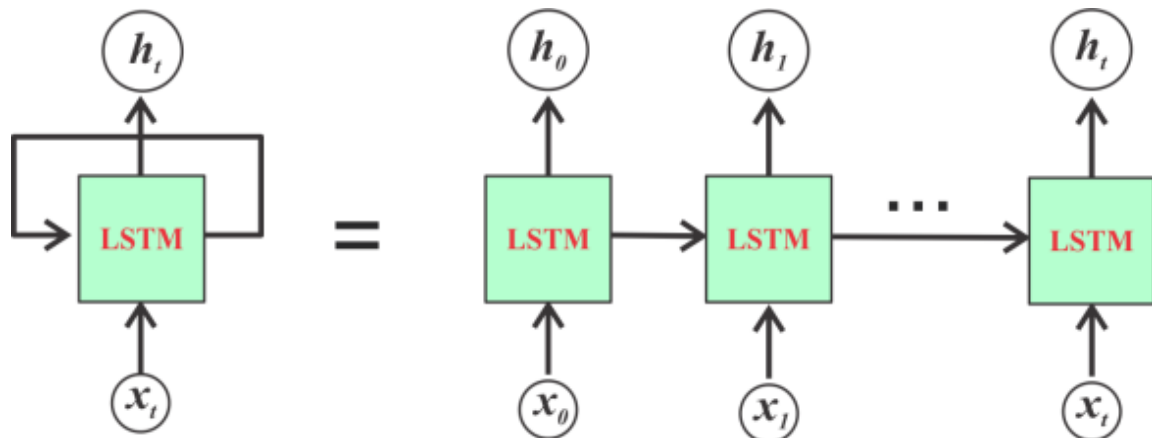
Layer (type)	Output Shape	Param #
=====		
dense (Dense)	(None, 120)	21480
-----		
dropout (Dropout)	(None, 120)	0
-----		
dense_1 (Dense)	(None, 80)	9680
-----		
dropout_1 (Dropout)	(None, 80)	0
-----		
dense_2 (Dense)	(None, 1)	81
=====		
Total params: 31,241		
Trainable params: 31,241		
Non-trainable params: 0		

---



### 3.5.2. LSTM:

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications.



```
model_1 = Sequential()
model_1.add(Bidirectional(LSTM(4, stateful=True, return_sequences=False), input_shape=(1,105000), batch_size=1))
model_1.add(Dense(1, activation='sigmoid'))
model_1.compile(loss='mse', optimizer='adam', metrics=['accuracy'])

model_1.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
=====		
bidirectional (Bidirectional)	(1, 8)	3360160
-----		
dense (Dense)	(1, 1)	9
=====		
Total params: 3,360,169		
Trainable params: 3,360,169		
Non-trainable params: 0		

loss: 0.1297 - accuracy: 0.8081 - val\_loss: 0.1481 - val\_accuracy: 0.8617

--

### 3.5.3. Sequential:

Sequential is the easiest way to build a model in Keras. It allows you to build a model layer by layer. Each layer has weights that correspond to the layer the follows it.

```
model_4 = Sequential()

model_4.add(Dense(100, input_dim=105000, activation='relu'))
model_4.add(BatchNormalization())
model_4.add(Dropout(0.2))

model_4.add(Dense(100, activation='relu'))
model_4.add(BatchNormalization())
model_4.add(Dropout(0.2))

model_4.add(Dense(1, activation='sigmoid'))

model_4.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model_4.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
=====	=====	=====
dense (Dense)	(None, 100)	10500100
batch_normalization (Batch Normalization)	(None, 100)	400
dropout (Dropout)	(None, 100)	0
dense_1 (Dense)	(None, 100)	10100
batch_normalization_1 (Batch Normalization)	(None, 100)	400
dropout_1 (Dropout)	(None, 100)	0
dense_2 (Dense)	(None, 1)	101
=====	=====	=====
Total params: 10,511,101		
Trainable params: 10,510,701		
Non-trainable params: 400		
=====		

loss: 0.1917 - accuracy: 0.9239 - val\_loss: 0.3739 - val\_accuracy: 0.8197

...

# Evaluation:

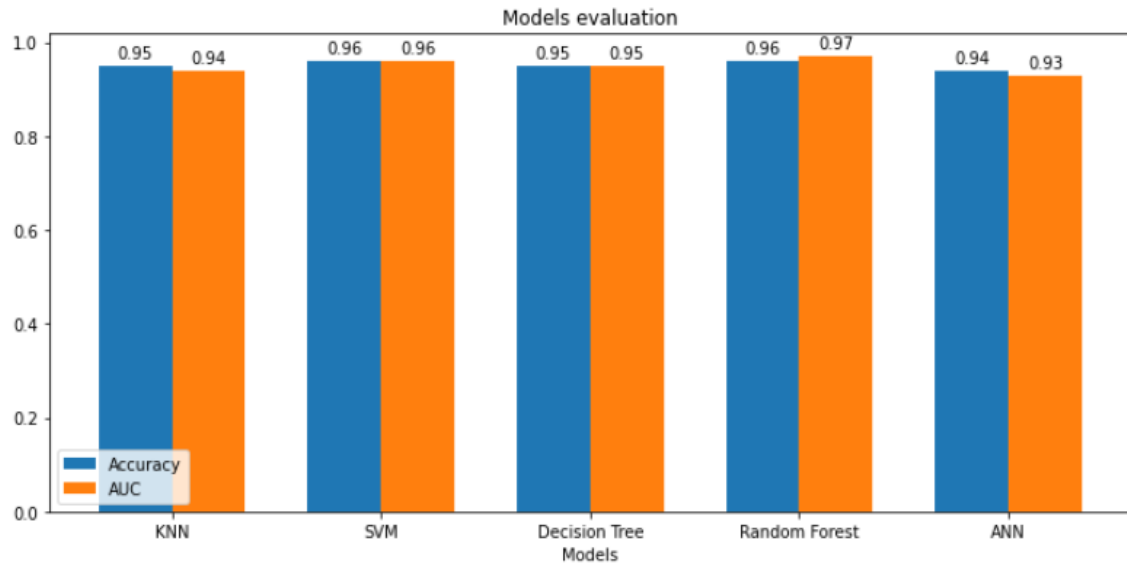
In the binary classification, as the labels are imbalanced, besides accuracy, we adopted area under the receiver operating characteristic (AUC) to evaluate the performance. In the multi-label classification, we employed precision on each class and an overall averaged accuracy to estimate our classifiers.

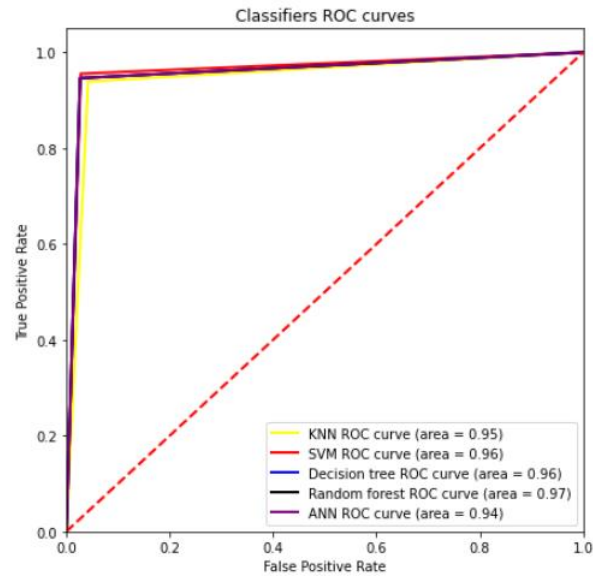
## 4.1. Tools:

Our pipeline was built on Python version 3.6.3. Machine learning classifiers, cross-validation, data pre-processing, parameters tuning was implemented by Scikit Learn package. We designed the architecture of deep neural networks with Keras. All experiments were deployed on Google CoLaboratory, which has dual CPU kernels and a GPU to expedite the computation time.

### 4.1.1. Bonn models evaluation:

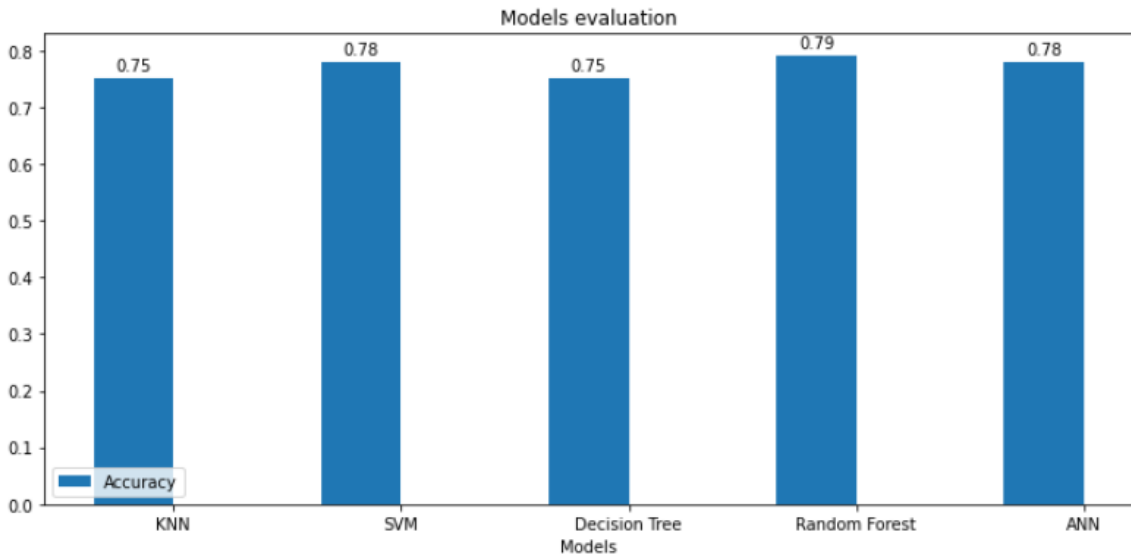
#### 4.1.1.1. Binary models(*Epileptic vs Healthy*):





For this binary classification all the models we applied have show good results, the random forest has the highest accuracy and AUC score but the large number of trees made the algorithm too slow so we can deploy SVM who showed good results also.

#### 4.1.1.2. Multiclass classification models (Seizure vs Tumor area vs Healthy area):



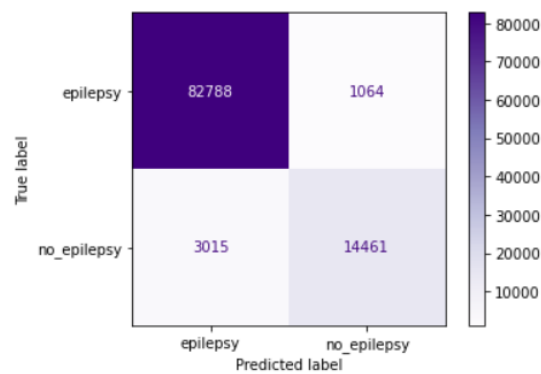
For this problem of multiclassification our only metric is the accuracy, random forest classifier has the highest accuracy but as we said in the previous section this classifier is too slow so for the deployment, we can choose the ANN or SVM who have equal accuracies.

## 4.1.2. Isip models evaluation:

### 4.1.2.1. Knn Model Evaluation:

```
[ ] print('Accuracy of K-NN classifier on training set: {:.2f}'  
      .format(knn.score(X_train, y_train)))  
print('Accuracy of K-NN classifier on test set: {:.2f}'  
      .format(knn.score(X_test, y_test)))
```

Accuracy of K-NN classifier on training set: 0.97  
Accuracy of K-NN classifier on test set: 0.96

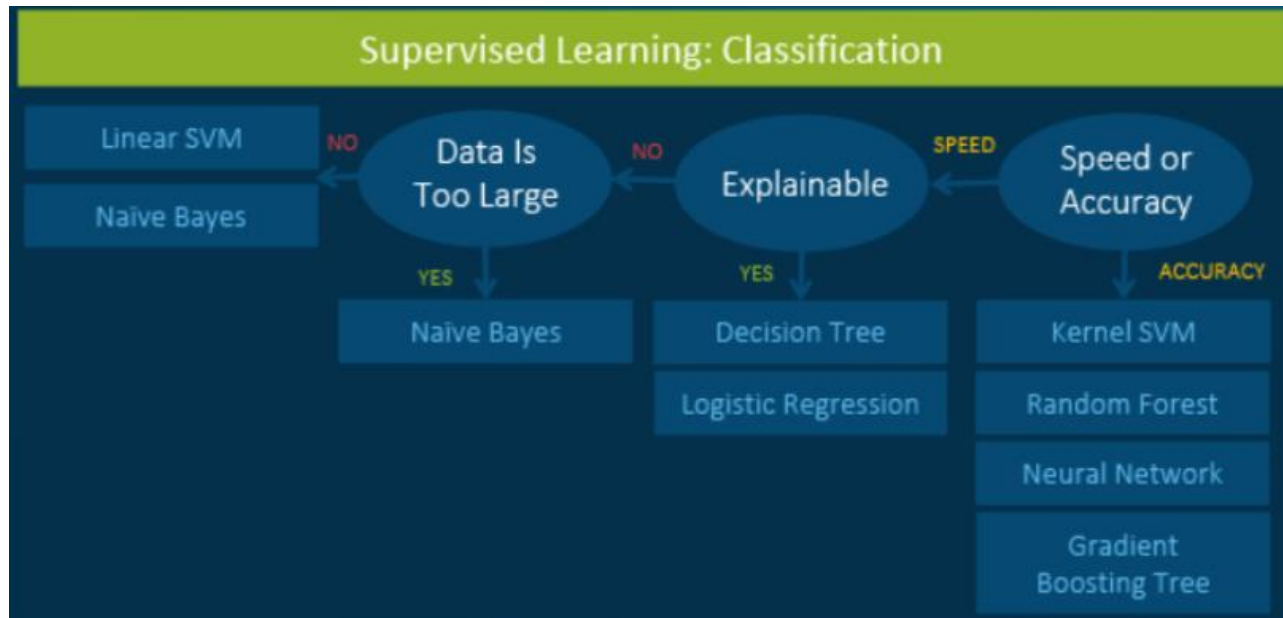


```
[ ] from sklearn.metrics import classification_report  
print(classification_report(y_test, y_pred_knn))
```

	precision	recall	f1-score	support
epilepsy	0.96	0.99	0.98	83852
no_epilepsy	0.93	0.83	0.88	17476
accuracy			0.96	101328
macro avg	0.95	0.91	0.93	101328
weighted avg	0.96	0.96	0.96	101328

### 4.1.3. External data models evaluation:

So, as we can see in the cheat sheet, Kernel SVM and Random Forest are the best algorithms if we want the best values of accuracy while Linear SVM and Naive Bayes are the best if we want fast results with both big and small data.



*Machine Learning supervised Learning classification algorithms cheat sheet*

And since we have few features in our data that are relevant and explainable we tried also to apply Decision Tree without forgetting to apply Grid Search Cross Validation to use the best parameters.

After applying the chosen algorithms, only SVM with sigmoid kernel gave a too low accuracy while all the other algorithms gave almost the same accuracy of 93% which is pretty good for a delicate field like HEALTH CARE.

Model	Score
Support Vector Machines RBF	0.933333
Support Vector Machines Linéaire	0.933333
Support Vector Machines polynomiale	0.933333
Support Vector Machines sigmoïde	0.200000

*FSVM Models' Accuracies with different kernels*

```
#Random Forest
```

```
model.score(X_test,y_test)
```

```
0.9333333333333333
```

*Random Forest Model Accuracy*

```
#DecisionTreeClassifier
```

```
print( 'le train_score=',final_model.score(X_train, y_train))
print( 'le test_score=',final_model.score(X_test, y_test))
```

```
le train_score= 1.0
le test_score= 0.9333333333333333
```

*Decision Tree Classifier Model Accuracy*

```
print("Meilleurs paramètres trouvés par GridSeachCV")
print(grid.best_params_)
```

```
Meilleurs paramètres trouvés par GridSeachCV
{'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}
```

*GridSearchCV*

Finally, we chose SVM with RBF kernel for deployment according to the GridSearch

## **Limitations and Future Work:**

Even though its remarkable outstanding performance in predicting seizures with the given test dataset, we cannot, currently, move into production since the real world can be extremely different than the theory in which this research was based on. Although, all what this model is being offered (real time sensors values, signals adjustment and doctors follow up) can lead it to promising results. Upgrading this research from theory to the practical world through an access to large medical records or via devices that can be given to epileptic people who will eventually volunteer in data collection for the sake of strengthening the brain's knowledge about Epilepsy and people who are suffering from it.



# Conclusion:

Our work demonstrates that machine learning classifiers and deep neural networks are useful in the recognition of Healthy and Epileptic patients, and Epileptic Seizure from EEG recording. Specifically, one ensemble classifier, random forest yields the best performance of both AUC and accuracy over 0.96. Deep neural networks significantly outperformed machine learning classifiers in the multi-label classification of brain activities. Our models showed potential usage in clinical decision making, such as identifying seizure in a timely manner.

Further study is needed to refine our networks to achieve a state-of-the-art result. A larger dataset is also important to validate the robustness of our models.

# References:

Bonn dataset:

[http://epileptologie-bonn.de/cms/front\\_content.php?idcat=193&lang=3](http://epileptologie-bonn.de/cms/front_content.php?idcat=193&lang=3)

Isip dataset: The TUH EEG Epilepsy Corpus (TUEP)

[https://www.isip.piconepress.com/projects/tuh\\_eeg/html/downloads.shtml](https://www.isip.piconepress.com/projects/tuh_eeg/html/downloads.shtml)

TSFEL:

<https://tsfel.readthedocs.io/en/latest/index.html>

PyEEG:

<https://www.hindawi.com/journals/cin/2011/406391/>