Automatic Seizure Detection Through Analysis of EEG Recordings Using Various Machine Learning Techniques

Erica Kok, Hala ElAarag

*College of Arts and Sciences, **Stetson University***

Contact email: ericakok28@gmail.com

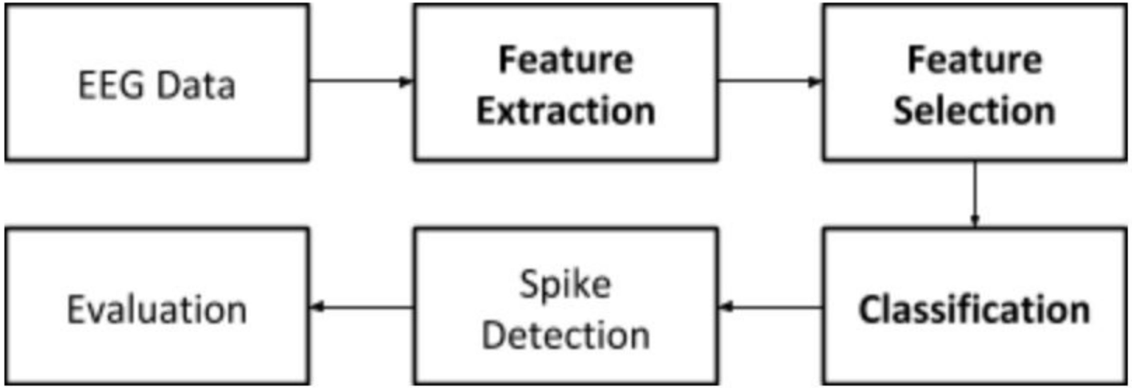
IARIA

Purpose of this research

- **Automate the process of detecting seizures + aid in the diagnosis of epilepsy**
    - Automatically label recordings in order to be able to diagnose, monitor, and plan patient treatment
    - Replace the need for laborious visual analysis of day-long recordings
    - Improve detection accuracy by decreasing the possibility of human error

Methodology

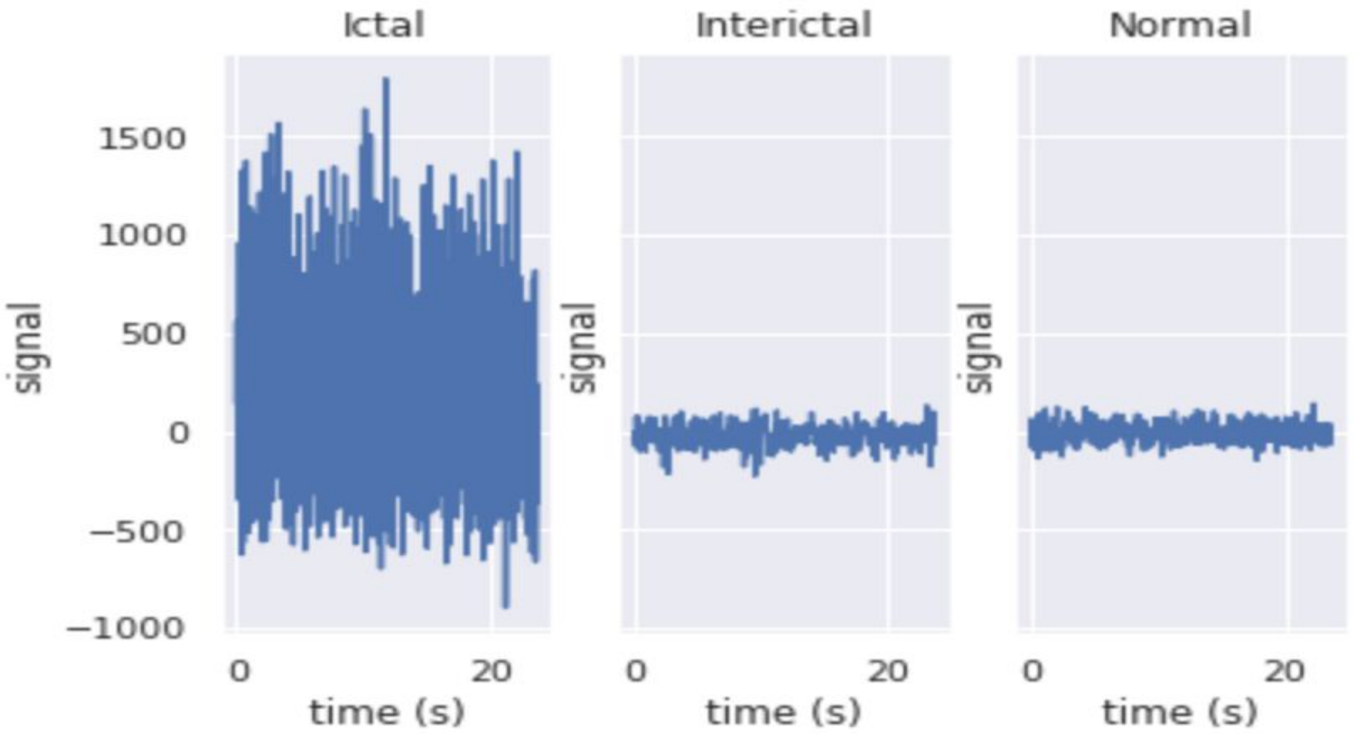
We need to determine when spikes and sharp waves occur in any given electroencephalogram (EEG) recording.



EEG Data

- University of Bonn dataset acquired by Andrzejak et al.
- Comprised of five datasets A, B, C, D, and E
- Each contains 100 single-channel EEG segments with a duration of 23.6 seconds
- Each dataset can be downloaded as a .zip file containing 100 .txt files
- Each .txt file consists of 4096 samples of one EEG time series in ASCII code

Data Processing



Feature Extraction

- Mean
- Median
- Amplitude
- Maximum
- Minimum
- Standard deviation
- Skewness
- Variance
- Energy of the signal
- Curve length of the signal

Feature Selection

- Perfectly correlated variables are truly redundant because there is no additional information that can be gained by adding them.
- Using a Pearson correlation matrix to select relevant variables

Classification

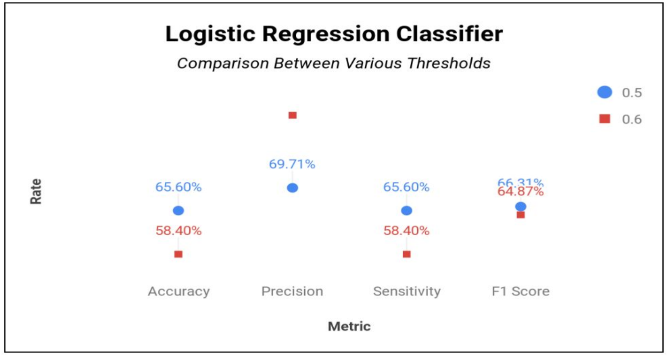
- Determine a boundary between the classes and label them based on their measured features
- Five categories of classification techniques:
  1. Linear classifiers
  2. Nonlinear Bayesian classifiers
  3. Nearest neighbor classifiers
  4. Ensemble classifiers
  5. Neural network

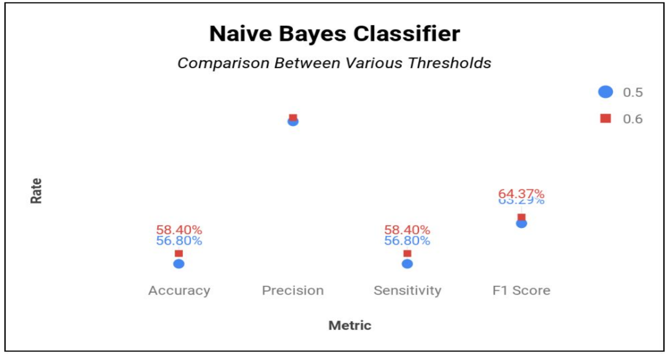
Evaluation

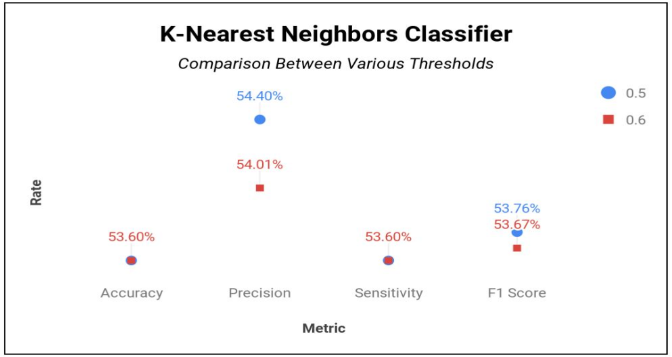
- The decision made by the classifier can be divided into four categories:

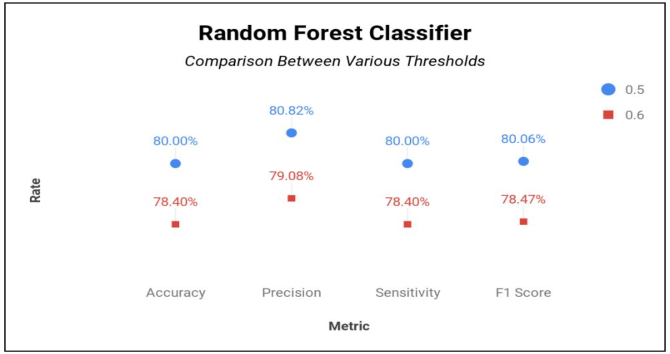
		Predicted class	
		P	*N*
Actual Class	*P*	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Summary of Initial Results

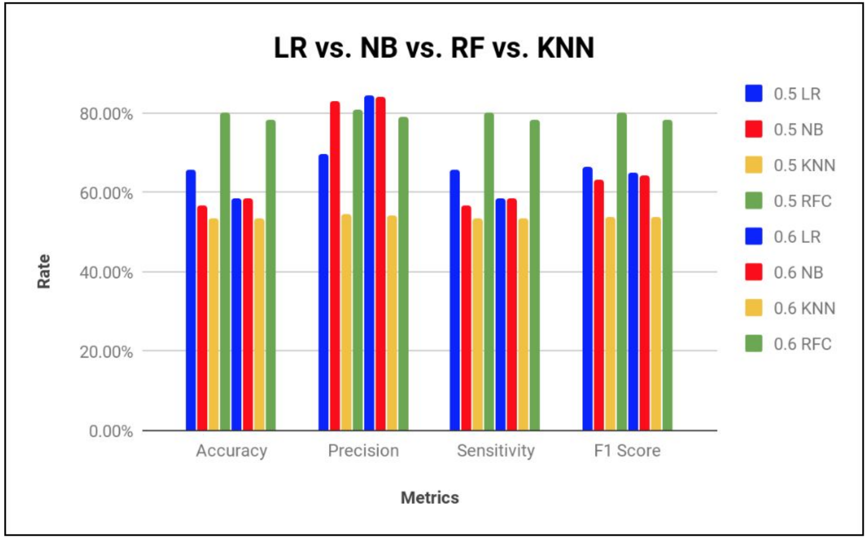








Summary of Initial Results



RFC with 0.5 threshold performed the best

A few issues with this method

- Selecting dataset
- Determining features to extract
- Overfitting models

Suggested improvements

- Use tsfresh to extract more features
- Use neural network for classification
- Compare different sizes of training data
- Utilize another dataset

Feature Extraction & Selection

- Used a Python package called Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (**tsfresh**)
  - Automates and accelerates the process of feature extraction and feature selection *by combining various characterization methods* with feature selection
  - **Highly parallel** feature selection algorithm based on statistical hypothesis tests
  - Automatically configured based on the type of supervised machine learning problem (classification/ regression) and the type of features (categorical/ continuous)

Classification

- Built LR, NB, KNN, and RFC models using *scikit-learn*
- Built a <u>Neural Network</u> using **Keras**
  - Added multiple **Dense layers** to model *(A layer of neurons in a neural network model where each neuron receives its input from the neurons in the previous layer)*
- Used **k-fold cross validation** technique with *5 splits*

Evaluation

- Utilized *seaborn's* **heatmap**

1D Convolutional Neural Network

- *Without* going through the process of feature extraction and feature selection (just <u>classification</u>)
- **First 1D CNN layer**
  - Kernel size of height 100 allows the neural network to learn one single feature in the first layer
  - Defined *32 filters* which allows us to train 32 different features on the first layer of the network
  - Output is a 3899 x 32 neuron matrix
  - Each column of the output matrix holds the weights of a single filter

Layer (type)	Output Shape	Param #
conv1d_270 (Conv1D)	(None, 3998, 32)	3232
conv1d_271 (Conv1D)	(None, 3899, 32)	102432
conv1d_272 (Conv1D)	(None, 3800, 16)	51216
max_pooling1d_148 (MaxPoolin	(None, 38, 16)	0
flatten_249 (Flatten)	(None, 608)	0
dense_583 (Dense)	(None, 32)	19488
dense_584 (Dense)	(None, 3)	99

Total params: 176,467
Trainable params: 176,467
Non-trainable params: 0

```
model_1d_cnn = Sequential([
    Conv1D(filters=32, kernel_size=100, input_shape=(4097,1)),
    Conv1D(filters=32, kernel_size=100, activation='relu'),
    Conv1D(filters=16, kernel_size=100, activation='relu'),
    MaxPooling1D(pool_size=100),
    Flatten(),
    Dense(32, activation='relu'),
    Dense(3, activation='softmax')
    ])
```

1D Convolutional Neural Network

- **Second + third 1D CNN layer**
  - Result from the previous CNN layer will be fed into the next CNN layer
  - Again defined 32 different filters to be trained on the 2nd level
  - Following the same logic as the previous layer, the output matrix from this layer will be of size 3800 x 16
  - Result from the second CNN layer will be fed into the third CNN layer, which outputs a matrix of size 38 x 16

Layer (type)	Output Shape	Param #
conv1d_270 (Conv1D)	(None, 3998, 32)	3232
conv1d_271 (Conv1D)	(None, 3899, 32)	102432
conv1d_272 (Conv1D)	(None, 3800, 16)	51216
max_pooling1d_148 (MaxPoolin	(None, 38, 16)	0
flatten_249 (Flatten)	(None, 608)	0
dense_583 (Dense)	(None, 32)	19488
dense_584 (Dense)	(None, 3)	99

Total params: 176,467
Trainable params: 176,467
Non-trainable params: 0

```
model_1d_cnn = Sequential([
    Conv1D(filters=32, kernel_size=100, input_shape=(4097,1)),
    Conv1D(filters=32, kernel_size=100, activation='relu'),
    Conv1D(filters=16, kernel_size=100, activation='relu'),
    MaxPooling1D(pool_size=100),
    Flatten(),
    Dense(32, activation='relu'),
    Dense(3, activation='softmax')
    ])
```

1D Convolutional Neural Network

- **Max pooling layer**
  - Added a max pooling layer of size 100 to reduce the complexity of the output and prevent overfitting of the data
  - Pool size of 100 means that the size of the output matrix of this layer is only a hundreth of the input matrix
- **Flatten layer**
  - Added a flatten layer to flatten the input before finally adding two dense layers

```
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_270 (Conv1D)          (None, 3998, 32)          3232
_________________________________________________________________
conv1d_271 (Conv1D)          (None, 3899, 32)          102432
_________________________________________________________________
conv1d_272 (Conv1D)          (None, 3800, 16)          51216
_________________________________________________________________
max_pooling1d_148 (MaxPoolin (None, 38, 16)            0
_________________________________________________________________
flatten_249 (Flatten)        (None, 608)               0
_________________________________________________________________
dense_583 (Dense)            (None, 32)                19488
_________________________________________________________________
dense_584 (Dense)            (None, 3)                 99
=================================================================
Total params: 176,467
Trainable params: 176,467
Non-trainable params: 0
```

```
model_1d_cnn = Sequential([
    Conv1D(filters=32, kernel_size=100, input_shape=(4097,1)),
    Conv1D(filters=32, kernel_size=100, activation='relu'),
    Conv1D(filters=16, kernel_size=100, activation='relu'),
    MaxPooling1D(pool_size=100),
    Flatten(),
    Dense(32, activation='relu'),
    Dense(3, activation='softmax')
    ])
```

1D Convolutional Neural Network

Dense Layers

- Reduces the vector of height 32 to a vector of 3 since we have 3 classes that we want to predict (Normal, Interictal, Ictal)
- Reduction is done by another matrix multiplication, we used softmax as the activation function
- Forces all 3 outputs of the neural network to sum up to one
- Final output value represents the probability for each of the 3 classes

Layer (type)	Output Shape	Param #
conv1d_270 (Conv1D)	(None, 3998, 32)	3232
conv1d_271 (Conv1D)	(None, 3899, 32)	102432
conv1d_272 (Conv1D)	(None, 3800, 16)	51216
max_pooling1d_148 (MaxPoolin	(None, 38, 16)	0
flatten_249 (Flatten)	(None, 608)	0
dense_583 (Dense)	(None, 32)	19488
dense_584 (Dense)	(None, 3)	99

Total params: 176,467
Trainable params: 176,467
Non-trainable params: 0

```
model_1d_cnn = Sequential([
    Conv1D(filters=32, kernel_size=100, input_shape=(4097,1)),
    Conv1D(filters=32, kernel_size=100, activation='relu'),
    Conv1D(filters=16, kernel_size=100, activation='relu'),
    MaxPooling1D(pool_size=100),
    Flatten(),
    Dense(32, activation='relu'),
    Dense(3, activation='softmax')
    ])
```

Summary of Results

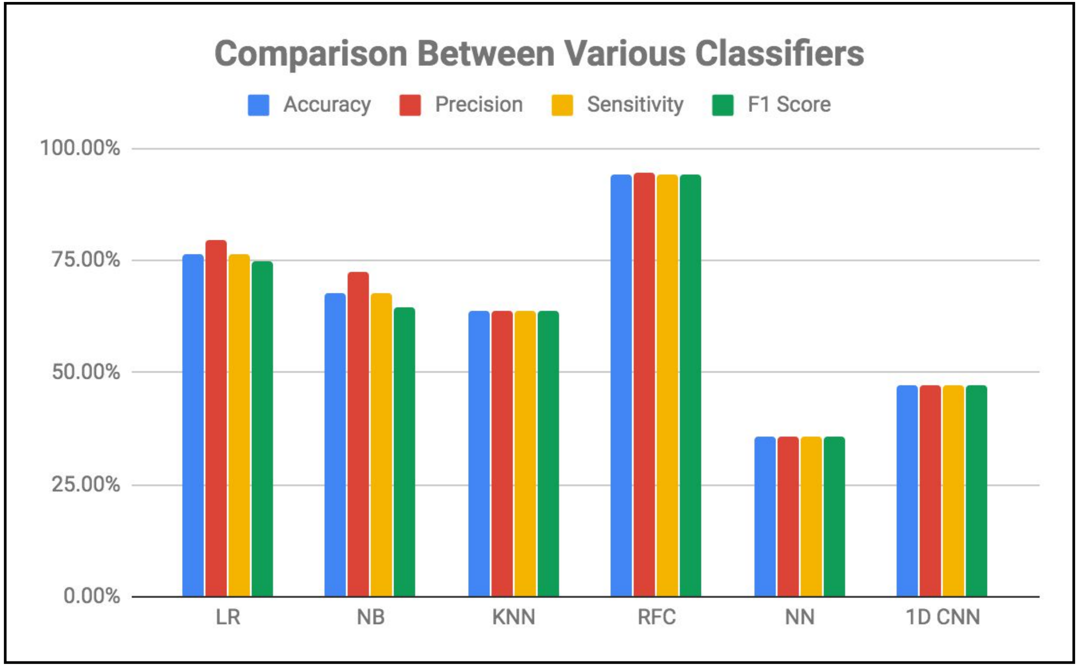
Neural Network

Accuracy	35.80%
Precision	35.80%
Sensitivity	35.80%
F1 Score	35.80%

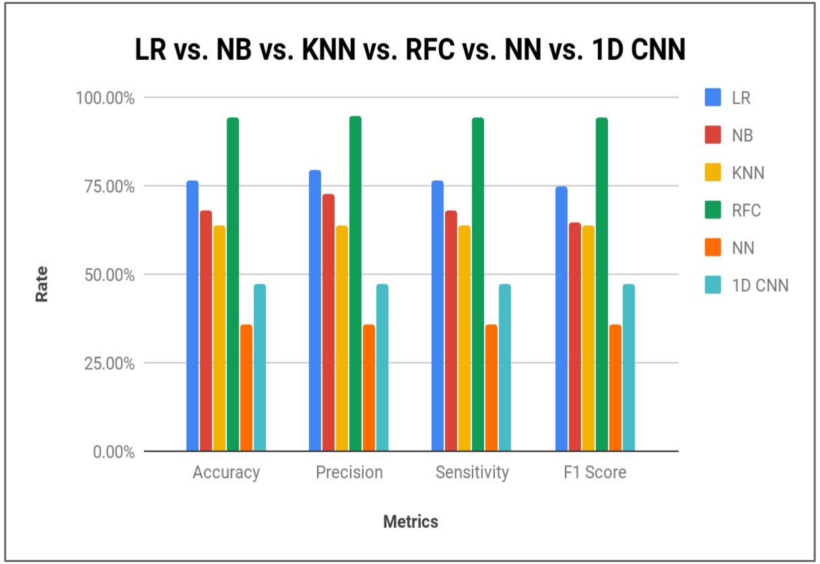
1D CNN

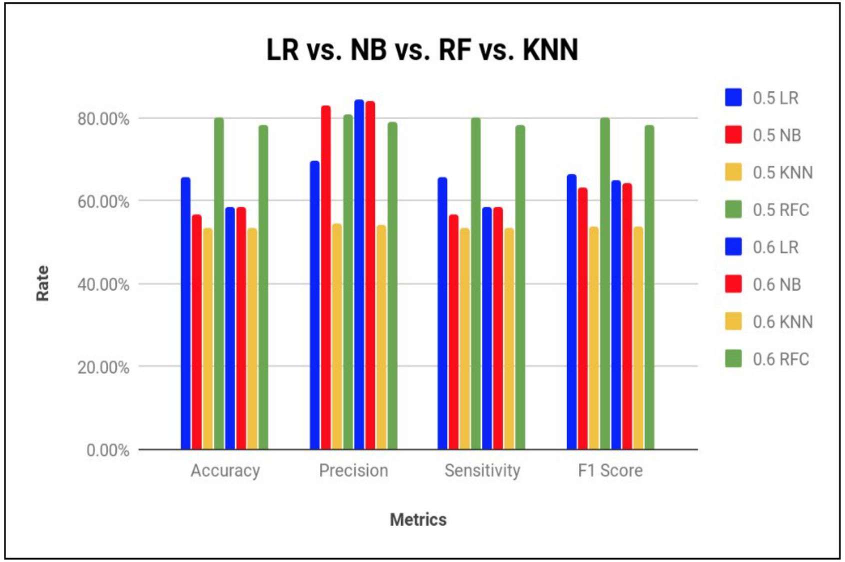
Accuracy	47.20%
Precision	47.20%
Sensitivity	47.20%
F1 Score	47.20%

Summary of Results



Comparison to Initial Results





Conclusion

- Automate seizure detection process
- Feature Extraction + Feature Selection
- 5 Classification Techniques + *k-fold cross validation*
- 4 Evaluation metrics
- **RFC** performed the best in all metrics
- **NN** performed the worst

Thank you!

Questions?