ML Algorithms for Diagnosis of Epileptic Seizures

submitted by
ARPITA GAUTAM (ME20BTECH11008)

for the course project of
Data Science Analysis (EP4130)

to
Dr. Shantanu Desai
Department of Physics

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Table of Contents

References

Data:

➢ https://archive.ics.uci.edu/ml/datasets/Epileptic+Seizure+Recognition

Papers:

➢ Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: dependence on recording region and brain state. R G Andrzejak, K Lehnertz, F Mormann, C Rieke, P David, C E Elger.
DOI: 10.1103/PhysRevE.64.061907
➢ A One-Dimensional CNN-LSTM Model for Epileptic Seizure Recognition Using EEG Signal Analysis.
https://doi.org/10.3389/fnins.2020.578126

Kaggle Notebooks:

➢ https://www.kaggle.com/code/abdelmalekelmoukhtar/ml-algorithms-for-epileptic-seizures/notebook
➢ https://www.kaggle.com/code/tonyfischer/epileptic-seizure-recognition#About-the-Dataset

GitHub Repository

The code used for this project can be found at following link:

https://github.com/ArpitaGautam8/DSA_Project_EP4130

Abstract:

The aim of this study is to diagnose epileptic seizures by using different machine learning algorithms. For this purpose, the frequency components of the EEG are extracted by using the Discrete Wavelet Transform (DWT) and parametric methods based on autoregressive (AR) model. Both these two feature extraction methods are applied to the input of machine learning classification algorithms such as Artificial Neural Networks (ANN), Naive Bayesian, k-Nearest Neighbor (k-NN), Support Vector Machines (SVM), Logistic Regression, Principal Component Analysis. The results show that k-NN, ANN and SVM were the most efficient method according to test processing of both DWT and AR as feature extraction for recognition of epileptic seizures in EEG.

1. Introduction

1.1 Epilepsy

Epilepsy is a disorder of the brain characterized by repeated seizures. A seizure is usually defined as a sudden alteration of behavior due to a temporary change in the electrical functioning of the brain. Normally, the brain continuously generates tiny electrical impulses in an orderly pattern.

These impulses travel along neurons — the network of nerve cells in the brain — and throughout the whole body via chemical messengers called neurotransmitters.

In epilepsy the brain's electrical rhythms tend to become imbalanced, resulting in recurrent seizures. In patients with seizures, the normal electrical pattern is disrupted by sudden and synchronized bursts of electrical energy that may briefly affect their consciousness, movements, or sensations.
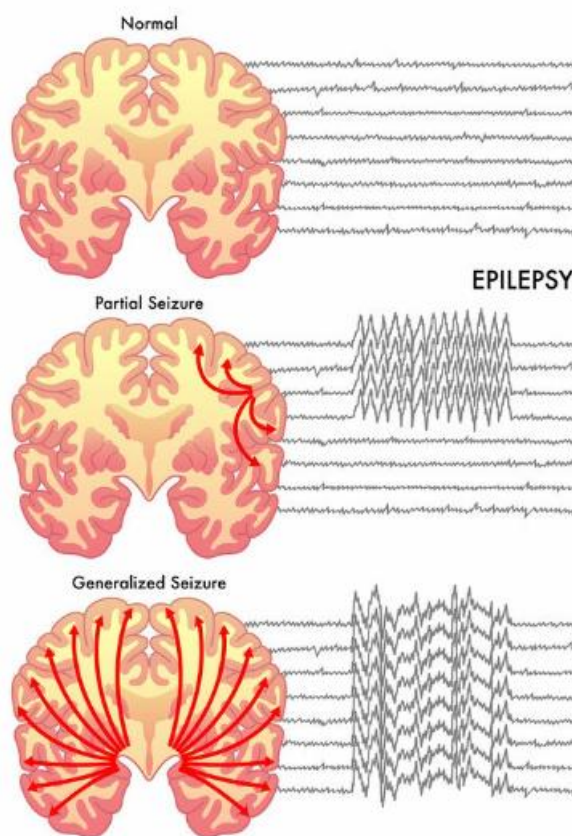
1.2 Diagnosis

Epilepsy is usually diagnosed after a person has had at least two seizures that were not caused by some known medical condition, such as alcohol withdrawal or extremely low blood sugar.

If seizures arise from a specific area of the brain, then the initial symptoms of the seizure often reflect the functions of that area. The right half of the brain controls the left side of the body, and the left half of the brain controls the right side of the body. For example, if a seizure starts from the right side of the brain in the area that controls movement in the thumb, then the seizure may begin with jerking of the left thumb or hand.

Diagnostic tests may include:

- **Blood tests**.
- **Electroencephalogram (EEG)** - a procedure that records the brain's continuous electrical activity by means of electrodes attached to the scalp.
- **Magnetic resonance imaging (MRI)** - a diagnostic procedure that uses a combination of large magnets, radiofrequencies, and a computer to produce detailed images of organs and structures within the body.
- **Computed tomography scan** (Also called a CT or CAT scan.) - a diagnostic imaging procedure that uses a combination of X-rays and computer technology to produce horizontal, or axial, images (often called slices) of the body. A CT scan shows detailed images of any part of the body, including the bones, muscles, fat, and organs. CT scans are more detailed than general X-rays.
- **Lumbar puncture (spinal tap)** - a special needle is placed into the lower back, into the spinal canal. This is the area around the spinal cord. The pressure in the spinal canal and brain can then be measured. A small amount of cerebral spinal fluid (CSF) can be removed and sent for testing to determine if there is an infection or other problems. CSF is the fluid that bathes your child's brain and spinal cord.

2. Theoretical Background

The aim of this study is to contribute to the diagnosis of epilepsy by taking advantage of engineering. So, for diagnosing of epileptic seizures from EEG signals are transformed discrete wavelet and auto regressive models. After these transformations, extracted data is applied as input for Naive Bayesian, k-Nearest Neighbor (k-NN), Support Vector Machines (SVM), ANN, Logistic Regression and Principal Component Analysis algorithms.

3. EEG Data Recording

EEG signals are separated into α, β, δ and θ spectral components and provide a wide range of frequency components. EEG spectrum contains some characteristic waveforms that fall primarily within four frequency bands as follows:

δ (0.5-4 Hz), θ (4-8 Hz), α (8-13 Hz), and β (13- 30 Hz).

EEG data set has acquired different age groups in this study. They are known as epileptic with uncontrolled seizures and are admitted to the neurology department of the Medical Faculty Hospital of Dicle University.

The EEG data used is collected from 400 people, 200 of them have epilepsy and 200 of them are normal. The EEG signals are contained by PCI-MIO 16E DAQ card system that provides real time processing and is a data bus of computer, signal processor and personal computer. Fig. 1 shows how to acquire EEG data from a patient.

EEG signals are to ensure the accuracy of diagnosing disease that usually takes 8-10 hours in the form of records. EEG signals are used in section and 23.6 seconds, 173 Hz sampling frequency is illustrated with. International 10–20 electrode placement system according to the data collected, 12-bit analog-digital conversion after the samples are recorded subsequently. Data can be passed through the filter 0.53–40 Hz band–pass, the EEG in the presence of clinical interest for focusing range is provided.

The EEG data used in our study were downloaded from 24-h EEG recorded from both epileptic patients and normal subjects. The following bipolar EEG channels were selected for analysis: F7-C3, F8-C4, T5-O1 and T6-O2. To assess the performance of the classifier, we selected 500 EEG segments containing spike and wave complex, artifacts, and background normal EEG.
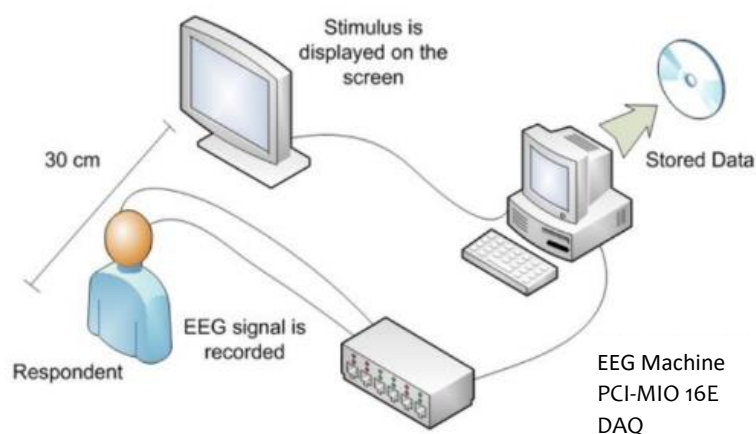
Fig.1 shows how EEG data is recorded.

Fig.1

4. Discrete Wavelet Transform (DWT)

Wavelet transform is a more advantageous spectral analyze method than other spectral analyze methods on non-stationary signals. Because the wavelet transform method changes large low-frequency, high frequency that is narrow for the window size. So, the entire frequency range can be achieved in the optimum time-frequency resolution. Continuous and discrete wavelet transform is analyzed in the scale and variation of parameters due to the continuous wavelet coefficients for each scale is difficult and time consuming. For this reason, discrete wavelet transform is used more often than these non-stationary signals. Wavelet scale is divided into several points for x[n] process as seen in Fig. 2 that is called "Multi Resolution Decomposition". Wavelet coefficients contain important information about EEG signal that provide extraction of feature vector. Statistical-time frequency of EEG signals sequences are:

1. The average of the absolute value of coefficients in each sub-band.
2. The maximum absolute value of coefficients in each sub-band.
3. The mean force coefficients of each sub-band.
4. Standard deviation of coefficients in each sub-band.
5. The average absolute value of the ratio of adjacent bands.
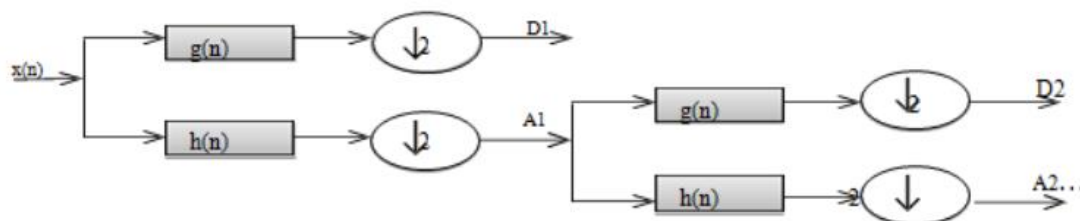6. Distribution of breakdown coefficients in each sub-band.

Fig. 2. Realization of discrete wavelet decomposition sub-bands; g[n] is high-pass filter, h[n] is low pass filter [22]

1-3 sequence is signal characteristic.

4-6 sequence is that amount of frequency change.

This feature vector, of EEG signals are used as inputs for multi-layer neural network classification.

5. The Data

5.1 Importing the libraries

```
[3]   1    import numpy as np
      2    import matplotlib.pyplot as plt
      3    import pandas as pd
      4    import seaborn as sn
```

5.2 Loading the Dataset

```
[48]  1    ESR = pd.read_csv("/content/Epileptic Seizure Recognition.csv")
      2
```

5.3 Showing the Dataset

```
1    ESR.head()
```

	Unnamed	X1	X2	X3	X4	X5	X6	X7	X8	X9	...	X170	X171	X172	X173	X174	X175	X176	X177	X178	y
0	X21.V1.791	135	190	229	223	192	125	55	-9	-33	...	-17	-15	-31	-77	-103	-127	-116	-83	-51	0
1	X15.V1.924	386	382	356	331	320	315	307	272	244	...	164	150	146	152	157	156	154	143	129	1
2	X8.V1.1	-32	-39	-47	-37	-32	-36	-57	-73	-85	...	57	64	48	19	-12	-30	-35	-35	-36	0
3	X16.V1.60	-105	-101	-96	-92	-89	-95	-102	-100	-87	...	-82	-81	-80	-77	-85	-77	-72	-69	-65	0
4	X20.V1.54	-9	-65	-98	-102	-78	-48	-16	0	-21	...	4	2	-12	-32	-41	-65	-83	-89	-73	0

5 rows × 180 columns

The original dataset from the reference consists of 5 different folders, each with 100 files, with each file representing a single subject/person. Each file is a recording of brain activity for 23.6 seconds.

The corresponding time-series is sampled into 4097 data points. Each data point is the value of the EEG recording at a different point in time. So, we have a total of 500 individuals with each having 4097 data points for 23.5 seconds.

We divided and shuffled every 4097 data points into 23 chunks, each chunk contains 178 data points for 1 second, and each data point is the value of the EEG recording at a different point in time.

So now we have 23 x 500 = 11500 pieces of information(row), each information contains 178 data points for 1 second(column), the last column represents the label y {1,2,3,4,5}.

y contains the category of the 178-dimensional input vector. Specifically, y in {1, 2, 3, 4, 5}:

5 - eyes open, means when they were recording the EEG signal of the brain the patient had their eyes open.

4 - eyes closed, means when they were recording the EEG signal the patient had their eyes closed.

3 - Yes, they identify where the region of the tumor was in the brain and recording the EEG activity from the healthy brain area.

2 - They recorded the EEG from the area where the tumor was located.

1 - Recording of seizure activity.

All subjects falling in classes 2, 3, 4, and 5 are subjects who did not have epileptic seizure. Only subjects in class 1 have epileptic seizures. Our motivation for creating this version of the data was to simplify access to the data via the creation of a .csv version of it. Although there are 5 classes, we have done binary classification, namely class 1 (Epileptic seizure) against the rest.

The response variable is y in column 179, the Explanatory variables X1, X2, ..., X178.

```
1   cols = ESR.columns
2   tgt = ESR.y.copy()   # Make a copy of the target variable
3   tgt[tgt > 1] = 0
4
5   # ax = sn.countplot(tgt, label="Count")
6   non_seizure, seizure = tgt.value_counts()
7   print('The number of trials for the non-seizure class is:', non_seizure)
8   print('The number of trials for the seizure class is:', seizure)
```

The number of trials for the non-seizure class is: 9200
The number of trials for the seizure class is: 2300
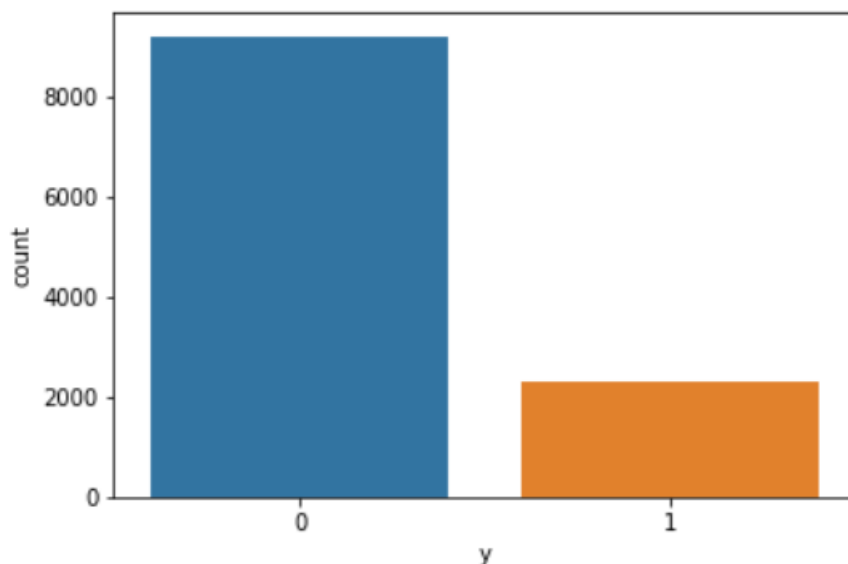
As we can see, there are 178 EEG features and 5 possible classes. The main goal of the dataset is to be able to correctly identify epileptic seizures from EEG data, so a binary classification between classes of label 1 and the rest (2,3,4,5). To train our model, let's define our independent variables (X) and our dependent variable (y).

6. Data Pre-processing

6.1 Checking the missing data:

```
[14]  1   ESR.isnull().sum()

      Unnamed     0
      X1          0
      X2          0
      X3          0
      X4          0
                 ..
      X175        0
      X176        0
      X177        0
      X178        0
      y           0
      Length: 180, dtype: int64
```

There is no missing value here, so we can work very smoothly.

7. Exploratory Data Analysis

In statistics, exploratory data analysis (EDA) is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task.

```
 1   ESR.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11500 entries, 0 to 11499
Columns: 180 entries, Unnamed to y
dtypes: int64(179), object(1)
memory usage: 15.8+ MB
```

```
 1   ESR.describe()
```

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	...	X170	
count	11500.000000	11500.000000	11500.000000	11500.000000	11500.000000	11500.000000	11500.000000	11500.00000	11500.00000	11500.000000	...	11500.000000	11500.00
mean	-11.581391	-10.911565	-10.187130	-9.143043	-8.009739	-7.003478	-6.502087	-6.68713	-6.55800	-6.168435	...	-10.145739	-11.63
std	165.626284	166.059609	163.524317	161.269041	160.998007	161.328725	161.467837	162.11912	162.03336	160.436352	...	164.652883	166.14
min	-1839.000000	-1838.000000	-1835.000000	-1845.000000	-1791.000000	-1757.000000	-1832.000000	-1778.00000	-1840.00000	-1867.000000	...	-1867.000000	-1865.00
25%	-54.000000	-55.000000	-54.000000	-54.000000	-54.000000	-54.000000	-54.000000	-55.00000	-55.00000	-54.000000	...	-55.000000	-56.00
50%	-8.000000	-8.000000	-7.000000	-8.000000	-8.000000	-8.000000	-8.000000	-8.00000	-7.00000	-7.000000	...	-9.000000	-10.00
75%	34.000000	35.000000	36.000000	36.000000	35.000000	36.000000	35.000000	36.00000	36.00000	35.250000	...	34.000000	34.00
max	1726.000000	1713.000000	1697.000000	1612.000000	1518.000000	1816.000000	2047.000000	2047.00000	2047.00000	2047.000000	...	1777.000000	1472.00

8 rows × 179 columns

Some samples:
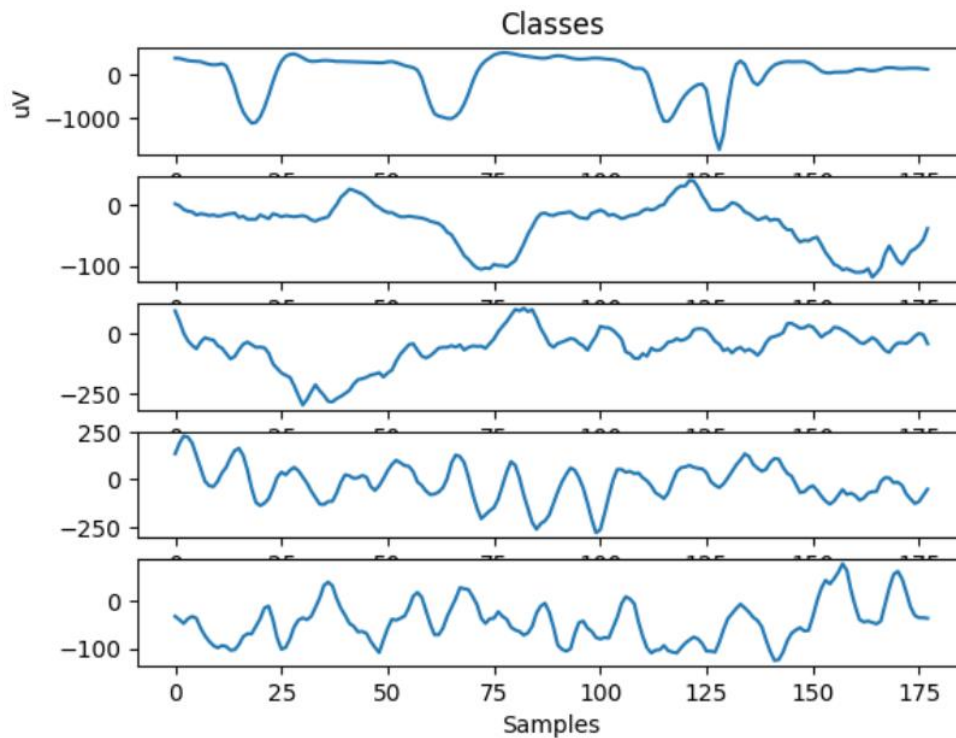
Classes

To make this a binary problem, we will make the non-seizure classes 0 while maintaining the seizure as 1.

```
[19]  1   y = ESR.iloc[:,179].values
      2   y
```

```
array([0, 1, 0, ..., 0, 0, 0])
```

```
      1   y[y>1]=0
      2   y
```

```
array([0, 1, 0, ..., 0, 0, 0])
```

8. Building ML models

Splitting Dataset into the Training and Test sets:

```
[21]  1   from sklearn.model_selection import train_test_split, cross_val_score
      2   X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

Feature Scaling:

```
[22]  1   from sklearn.preprocessing import StandardScaler
      2   sc = StandardScaler()
      3   X_train = sc.fit_transform(X_train)
      4   X_test = sc.transform(X_test)
```

8.1 Logistic Regression

Logistic regression, logic regression, or logit model is a regression model where the dependent variable (DV) is categorical. This article covers the case of a binary dependent variable—that is, where it can take only two values, "0" and "1", which represent outcomes such as pass/fail, win/lose, alive/dead or healthy/sick. Cases where the dependent variable has more than two outcome categories may be analyzed in multinomial logistic regression, or, if the multiple categories are ordered, in ordinal logistic regression.

```
[24]  1  from sklearn.linear_model import LogisticRegression
      2  clf = LogisticRegression()
      3  clf.fit(X_train, y_train)
      4  y_pred_log_reg = clf.predict(X_test)
      5  acc_log_reg = round(clf.score(X_train, y_train) * 100, 2)
      6  print (str(acc_log_reg) + ' %')
```

```
82.22 %
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

8.2 Support Vector Machine (SVM)

Support Vector Machine (SVM) model is a Supervised Learning model used for classification and regression analysis. It 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. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. Suppose some given data points each belong to one of two classes, and the goal is to decide which class a new data point will be in. In the case of support vector machines, a data point is viewed as a p-dimensional vector (a list of p numbers), and we want to know whether we can separate such points with a $(p-1)$-dimensional hyperplane.

When data is not labeled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups. The clustering algorithm which provides an improvement to the support vector machines is called support vector clustering and is often used in industrial applications either when data are not labeled or when only some data are labeled as a preprocessing for a classification pass.

In the below code, SVC stands for Support Vector Classification.

```
1   from sklearn.svm import SVC
2   clf = SVC()
3   clf.fit(X_train, y_train)
4   y_pred_svc = clf.predict(X_test)
5   acc_svc = round(clf.score(X_train, y_train) * 100, 2)
6   print (str(acc_svc) + '%')
```

98.22%

Linear SVM:

```
1   from sklearn.svm import SVC, LinearSVC
2   clf = LinearSVC()
3   clf.fit(X_train, y_train)
4   y_pred_linear_svc = clf.predict(X_test)
5   acc_linear_svc = round(clf.score(X_train, y_train) * 100, 2)
6   print ("Acc="+ str(acc_linear_svc) + '%')
```

Acc=81.99%
/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:1244: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
  warnings.warn(

8.3 K-Nearest Neighbors

k -nearest neighbors' algorithm (k-NN) is one of the simplest machine learning algorithms and is used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k -NN is used for classification or regression:

In k -NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k=1, then the object is simply assigned to the class of that single nearest neighbor.

In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors.

```
[27]  1   from sklearn.neighbors import KNeighborsClassifier
      2   clf = KNeighborsClassifier()
      3   clf.fit(X_train, y_train)
      4   y_pred_knn = clf.predict(X_test)
      5   acc_knn = round(clf.score(X_train, y_train) * 100, 2)
      6   print (str(acc_knn)+'%')
```

93.82%

8.4 Gaussian Naïve Bayes

Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

Bayes' theorem (alternatively Bayes' law or Bayes' rule) describes the probability of an event, based on prior knowledge of conditions that might be related to the event. For example, if cancer is related to age, then, using Bayes' theorem, a person's age can be used to more accurately assess the probability that they have cancer, compared to the assessment of the probability of cancer made without knowledge of the person's age.

```
[28]  1  from sklearn.naive_bayes import GaussianNB
      2  clf = GaussianNB()
      3  clf.fit(X_train, y_train)
      4  y_pred_gnb = clf.predict(X_test)
      5  acc_gnb = round(clf.score(X_train, y_train) * 100, 2)
      6  print (str(acc_gnb) + '%')
```

95.64%

8.5 Artificial Neural Network (ANN)

8.5.1   Modeling Artificial Neurons

Artificial neuron models are at their core simplified models based on biological neurons. This allows them to capture the essence of how a biological neuron functions. We usually refer to these artificial neurons as 'perceptrons'. So now let's look at what a perceptron looks like:
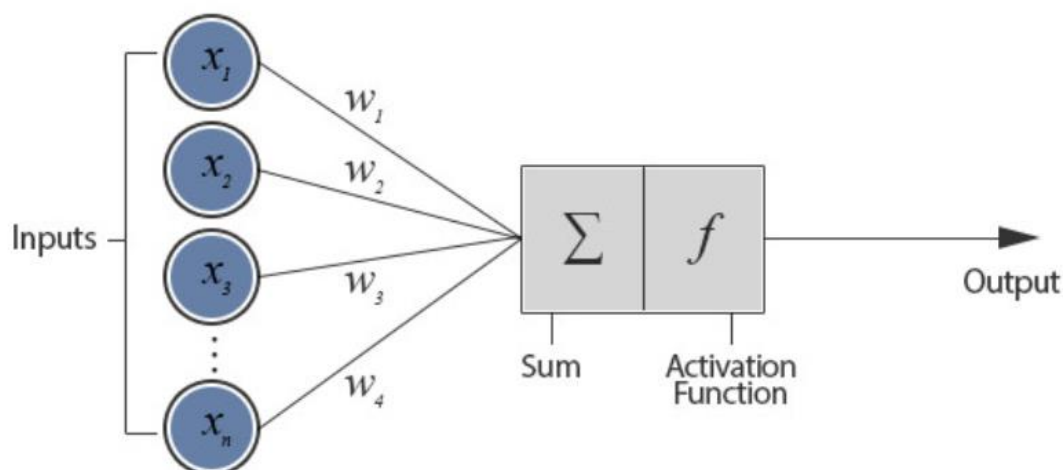

As shown in the diagram above, a typical perceptron will have many inputs and these inputs are all individually weighted. The perceptron weights can either amplify or reamplify the original input signal. For example, if the input is 1 and the input's weight is 0.2 the input will be decreased to 0.2. These weighted signals are then added together and passed into the

activation function. The activation function is used to convert the input into a more useful output. There are many different types of activation functions but one of the simplest would be the step function. A step function will typically output a 1 if the input is higher than a certain threshold, otherwise its output will be 0.

8.5.2   Implementing ANN:

Each input from the input layer is fed up with each node in the hidden layer, and from there to each node on the output layer. We should note that there can be any number of nodes per layer and there are usually multiple hidden layers to pass through before ultimately reaching the output layer. Choosing the right number of nodes and layers is important later on when optimizing the neural network to work well in a given problem. As you can probably tell from the diagram, it's called a feedforward network because of how the signals are passed through the layers of the neural network in a single direction. These aren't the only type of neural network though. There are also feedback networks where its architecture allows signals to travel in both directions.
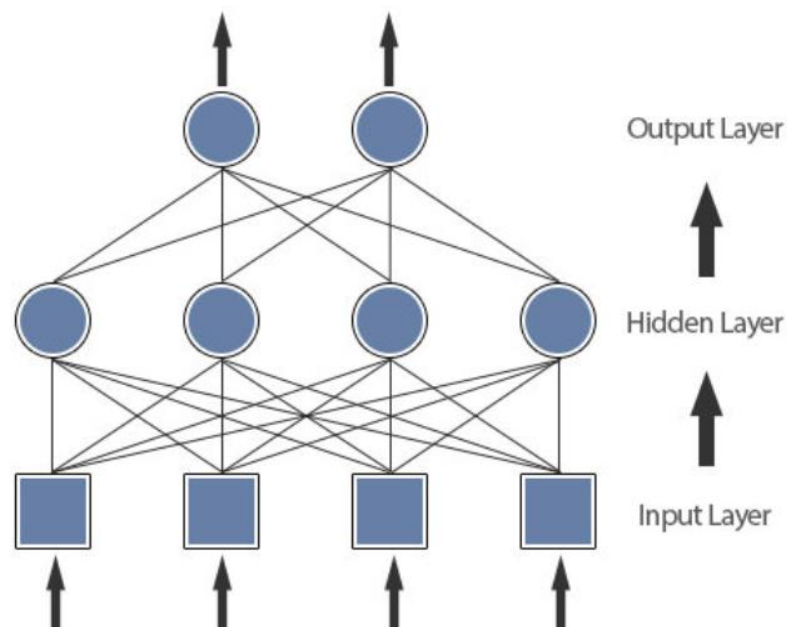
8.5.3   Initializing the ANN:

```
[29]   1    #Importing keras libraries and packages
       2    import keras
       3    from keras.models import Sequential
       4    from keras.layers import Dense
```

```
[30]   1    classifier = Sequential()
```

8.5.4   Adding input layer and first hidden layer:

```
[34]   1    classifier.add(Dense(units = 80,  activation = 'relu', input_dim = 178))
```

8.5.5 Adding second hidden layer:

```
[35]  1   classifier.add(Dense(units = 80,  activation = 'relu'))
```

8.5.6 Adding the output layer:

```
[36]  1   classifier.add(Dense(units = 1,  activation = 'sigmoid'))
```

8.5.7 Compiling the ANN:

```
[38]  1   classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
      2
      3   #Fitting the ANN to the training set
      4   classifier.fit(X_train, y_train, batch_size = 10)
```

```
920/920 [==============================] - 5s 3ms/step - loss: 0.2079 - accuracy: 0.9421
<keras.callbacks.History at 0x7fbfca015150>
```

```
[39]  1   # Predicting the Test set results
      2   y_pred = classifier.predict(X_test)
      3   y_pred = (y_pred > 0.5)
      4   acc_ANN = round(clf.score(X_train, y_train) * 100, 2)
      5   print (str(acc_ANN) + '%')
```

```
72/72 [==============================] - 0s 2ms/step
95.64%
```

8.6 Principal Component Analysis (PCA)

Principal Component Analysis (PCA)is a dimension-reduction tool that can be used to reduce a large set of variables to a small set that still contains most of the information in the large set.

How does PCA work -

1. Calculate the covariance matrix X of data points.
2. Calculate eigen vectors and corresponding eigen values.
3. Sort the eigen vectors according to their eigen values in decreasing order.
4. Choose the first k eigen vectors and that will be the new k dimensions.
5. Transform the original n dimensional data points into k dimensions.

```
[40]  1   from sklearn.decomposition import PCA
      2   pca = PCA()
      3   X_train_pca = pca.fit_transform(X_train)
      4   X_test_pca = pca.transform(X_test)
      5   acc_PCA = round(pca.score(X_train, y_train) )
      6   print (str(acc_PCA) + '%')
```

```
91%
```

9. Comparing Models

```python
models = pd.DataFrame({
    'Model': ['Logistic Regression', 'Support Vector Machines','ANN',
              'KNN', 'Naive Bayes', 'Principal Component Analysis'],

    'Score': [acc_log_reg, acc_svc,acc_ANN ,
              acc_knn, acc_gnb,acc_PCA ]
    })

models.sort_values(by='Score', ascending=False)
```

	Model	Score
1	Support Vector Machines	98.22
2	ANN	95.64
4	Naive Bayes	95.64
3	KNN	93.82
5	Principal Component Analysis	91.00
0	Logistic Regression	82.22

10.   Conclusion

We have classified epileptic seizures as opposed to normal seizures. Based on the model comparison, we can see that Support Vector Classification gives an accuracy of 98.22 %. Currently, medical experts recognize epileptic seizure activity through the visual inspection of electroencephalographic (EEG) signal recordings of patients based on their experience, which takes much time and effort. Hence, we can potentially use SVM Classification to automate the process thus getting results faster. There are a couple of limitations to the current approach, such as we need a large amount of labeled EEG signal data. However, collecting sufficient labeled data is time-consuming and laborious.

The ANN also provides us with a promising result by identifying correct epileptic seizures at the rate of 95.64%. The proposed model can be further improved and optimized to perform better on more challenging epileptic seizure recognition tasks, which will enhance its ability to classify data sets across different sets; second, the transfer learning technique can be added to the proposed model to lessen its reliance on labelled signal data. These two aspects will be the focus of future work.

Thank you.