MULTIVARIATE TIME SERIES PATTERN RECOGNITION USING MACHINE LEARNING AND DEEP LEARNING METHODS

Thesis

Submitted to the Graduate Faculty of

The University of Texas at Arlington

in partial fulfillment of the

requirements of the degree of

Master of Science

in

Industrial Engineering

By

Sai Abhishek Devar

B.Tech. Manipal Institute of Technology, 2016

December 2019

Acknowledgments

I would like to thank my advisor, Dr. Shouyi Wang, for his invaluable guidance and motivation throughout this work. I would also like to thank Dr. Kan Chen and Dr. Yuan Zhou for serving on my thesis defense committee. A special thanks to my family and friends for their unconditional love and support.

Table of Content

List of Figures

List of Tables

Abstract

In this research work, we have implemented machine learning & deep-learning algorithms on real-time multivariate time series datasets in the manufacturing & health care fields. The research work is organized into two case-studies.

The case study-1 is about rare event classification in multivariate time series in a pulp and paper manufacturing industry, data was collected of multiple sensors at each stage of the production line, the data contains a rare event of paper break that commonly occurs in the industry. For preprocessing we have implemented a sliding window approach for calculating the first-order difference method to capture the variation in the data over time. The sliding window approach helps to arrange the data for early prediction, for instance, we can set sliding window parameters to predict two or four minutes early as required. Our results indicate that for case study-1 best accuracy score was produced by the TensorFlow deep neural network model it was able to predict 50% of failures and 99% of non-failures with an overall accuracy of 75%.

In case study-2 we have brain EEG signal data of patients which were collected with the help of the Stereo EEG Implantation strategy to measure their ability to remember words shown to him/her after distracting him /her with math problems and other activities. The data was collected at a health-care lab at UT-Southwestern Medical Center. The brain EEG signal data collected by the company was preprocessed by using Pearson's and Spearman's correlations, extracting bandwidth frequencies and basic statistics from EEG signal data extracted for each event, event in case study-2 refers to a word shown to a patient. We have used minimum redundancy and maximum relevance feature selection method for dimensionality reduction of the data and to get the most effective features out of all. For case-study 2 best results were produced by SVM-RBF i.e. 73% accuracy to predict if a patient will remember or not remember a word.

Keywords: Multivariate time series, a rare event, machine learning, EEG-Signal, Sliding window, Deep-learning, Signal data, Sensor data.

Chapter-1

Introduction

Machine learning and artificial intelligence have been used in predicting and forecasting in various fields and have been providing some important insights in improving the performance of various industries. Though machine learning cannot replace a human in some of the complicated professions, it can help humans to be efficient in their work and guides them in a direction. In recent years machine learning is being successful in predicting stocks, sales, fraud detection, demand prediction, self-driven cars and disease classification [1][2]. Machine learning is still a developing field and making significant discoveries yet to this date [3]. Deep-learning is a subcategory of machine learning the main ideal behind deep-learning is to imitate human brain functionality [4]. Deep-learning neural network algorithms are very good in learning image data and very powerful tool of artificial intelligence devices such as self-driven cars [5], facial recognition, prediction of natural disasters, language translation, etc. A general deep learning neural network consists of an input layer, output layer and hidden layers with neurons. The major goal of machine-learning and deep-learning is to enable computers to make their own decisions in the absence of humans. A typical machine learning process can be providing data to a machine to train on by using various machine learning algorithms, tuning the parameters to optimize the performance of the model. The type of model used will depend on the type of data we are working on. In recent years we are generating a lot of signal data or sensor data that cannot be used directly to train a machine learning algorithm, so we need to follow some preprocessing and feature extraction to be applied on signal data to help the machines to detect patterns in our signal data.

1.1 Multivariate time series data

A general definition of multivariant time-series data would be data collected from multiple sources concerning time at equal intervals, each source depends on what happens in other sources. Currently, multivariant time series data is generated in every industry health care, manufacturing, sales & marketing, weather data and sensor data [6]. One example of multivariant time series data will be EEG-brain data, a typical EEG headset consists of several electrodes or also known as channels each channel collects data from different positions of the brain. Other examples can be installed multiple sensors in a production line these sensors can be measuring density, speed, power, etc. The data is collected by the sensors at equal time intervals. Stock market data is also a good example of multivariant time series data.

1.2 Literature Review of Machine-learning & Deep-learning

Machine learning enables machines to effectively interpret the data collected, for a human it is difficult to get insights by just viewing the data without performing any statistical analysis, in other words, human will take moderately high time for analyzing large data and it is tough to see patterns without performing an extensive analysis for a human without an involvement of a machine, to simplify this process machine learning is applied [7]. There have been many studies performed on how to teach a machine by themselves on the given datasets [8][9]. Machine learning was coming around when Alan Turing in the year 1950, created the Turing-Test for computers, this test aim to convince a human that he/she is talking to another human via text. In 1952, Arthur Samuel has created the first machine learning program which could learn how to play checkers. Neural-networks first came across in the year 1943 when Walter Pitts mathematician and

neurophysiologist Warren McCulloch implemented a mathematical model and algorithm inspired by the human neural network. In the year 1965 first functional deep-neural network was demonstrated by Alexey Ivakhnenko (Mathematician) and V.G, Lapa, they went on to implement a deep neural network with 8 layers in 1971. Kunihiko Fukushima designed an ANN (Artificial Neural Network) that was able to recognize visual patterns in 1979. Though neural networks were discovered early, the implementation required high computational power which made it hard to use frequently. Neural networks have started to get popular in the early 2000s as the rise of computational technology took place. Currently, leading companies like Google, Amazon, Facebook, OpenAI (Tesla) have invested in machine learning research in the early 21st century for improvising their research and sustain competition. In 2017, companies like google and amazon have started to renting GPU's through cloud computing, this is a cheaper alternative for buying high-quality GPU's and help in easy access of GPU's for people can lead to much more discoveries in the field of Deep-learning and Artificial-Neural-Networks. Deep-learning methods such as artificial neural networks, convolution neural networks & recurrent neural networks were inspired by the biological human neural system [10]. As we saw in then time-line neural networks were first implemented in 1943 it been more than 70 years they are around [11]. Neural network algorithms are now more effectively used for pattern recognition problems such as facial detection, tumor prediction, computer vision and many more [12][13]. Convolution neural networks is a powerful tool implemented in object detection and working with high dimensional data has been made easier, for example, data with images of objects, health care images (tumor, cancer), audio waveforms and they have been performing very effective when compared to traditional machine learning algorithms [14][15][16]. Other powerful models of deep learning such as recurrent neural network, transfer learning & encoding-decoding neural networks [14]. Studying these core

research, innovating our daily lives with tech and passion to implement learning were our main sources of motivation for this study.

Chapter-2

Rare event prediction in multivariate time series

2.1 Introduction to the problem statement

We are dealing with a real-time manufacturing dataset, as an industrial engineer our main goal is to produce high quality at a minimum cost, with the help of machine learning and artificial intelligence multiple aspects of manufacturing can be impacted such as improving plant performance, equipment maintenance, product quality, forecasting demand, reducing failures and employee's safety [17].

2.2 Objective of the study

A multivariate time series (MTS) has been generated when multiple interconnected streams of data are recorded over time. They are commonly found in manufacturing processes that have several interconnected sensors collecting the data over time. In this problem, we have a similar multivariate time series data from a pulp-and-paper industry with a rare event associated with them. It is an unwanted event in the process of a paper break, in our case that should be prevented.

2.3 Data Exploration

The dataset consists of 18,398 observations in total with columns timestamp, sixty-one columns sensor data each column represents sensor reading they can be the type of paper being processed, pulp type, blade speed, rotor speed, etc. For each timestamp, there is a binary variable that is represented as y is the label for every timestamp, zero being there was no rare event i.e. considered as production was going through and one being there is a paper break in the production cycle.

There are only 124 paper breaks out of 18,398 observations. This indicates that there is an imbalance in the data which will raise issues in training.

Here the sixty-one sensor columns will be our X variable i.e. our predictor variables and corresponding labels as y.

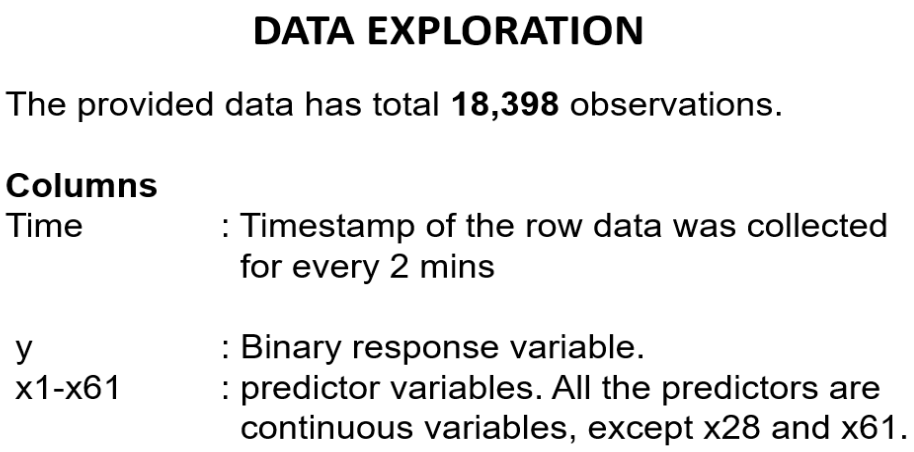


Figure 2.1: Data description

2.4 Data Visualization

Firstly, we are going to visualize if our data has a balance between classes/response variable to make sure if we have to perform any under or oversampling process.

From Figure 2.2, we know that there is a bias issue with the data as we have one of the class labels with only 124 observations and other class with 18,274 observations. This bias issue will be tackled in the preprocessing of the data.

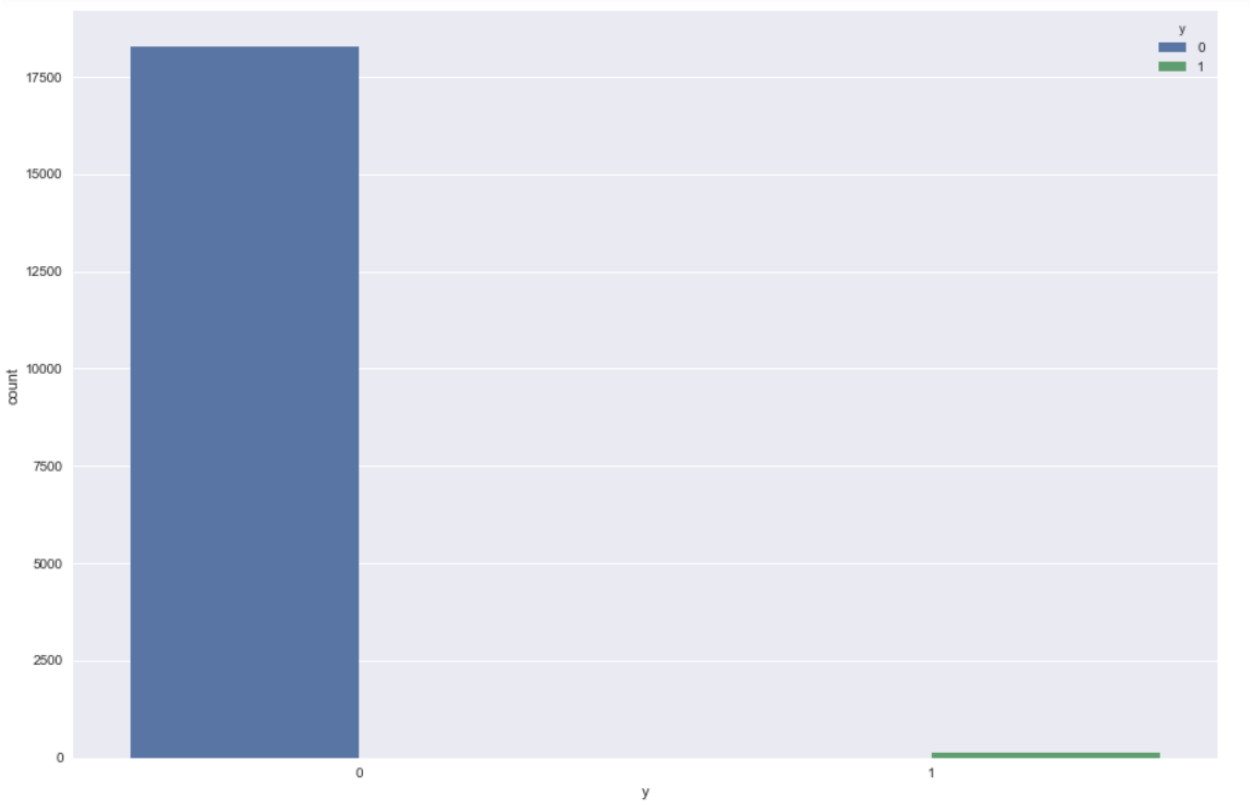


Figure 2.2: count of no of observations for each label

The next important visualization would be to check if we have any features are helping to predict y-variables. This can be done by checking the correlation of y-variable with any of our x-variables. By plotting the sixty-one x-variable features along with the time we can see in Figure 2.3 and Figure 2.4, there is a lot of noise and signal drift in the sensor data. By this, we have realized that our results may not be good and the classifications models will not be able to predict our minority class labels due to a major class imbalance if we train our models on default signal readings. One more limitation we need to consider would be we don't which x-variable is what, so we cannot just use the signal data as they are a need to do some feature extraction. In general, feature extraction is good for gaining information on signal patterns this method has proven to be good in many other explorations of data consisting of signal readings in the fields of health care such as analyzing heart/cardio signal, brain EEG signals, etc.

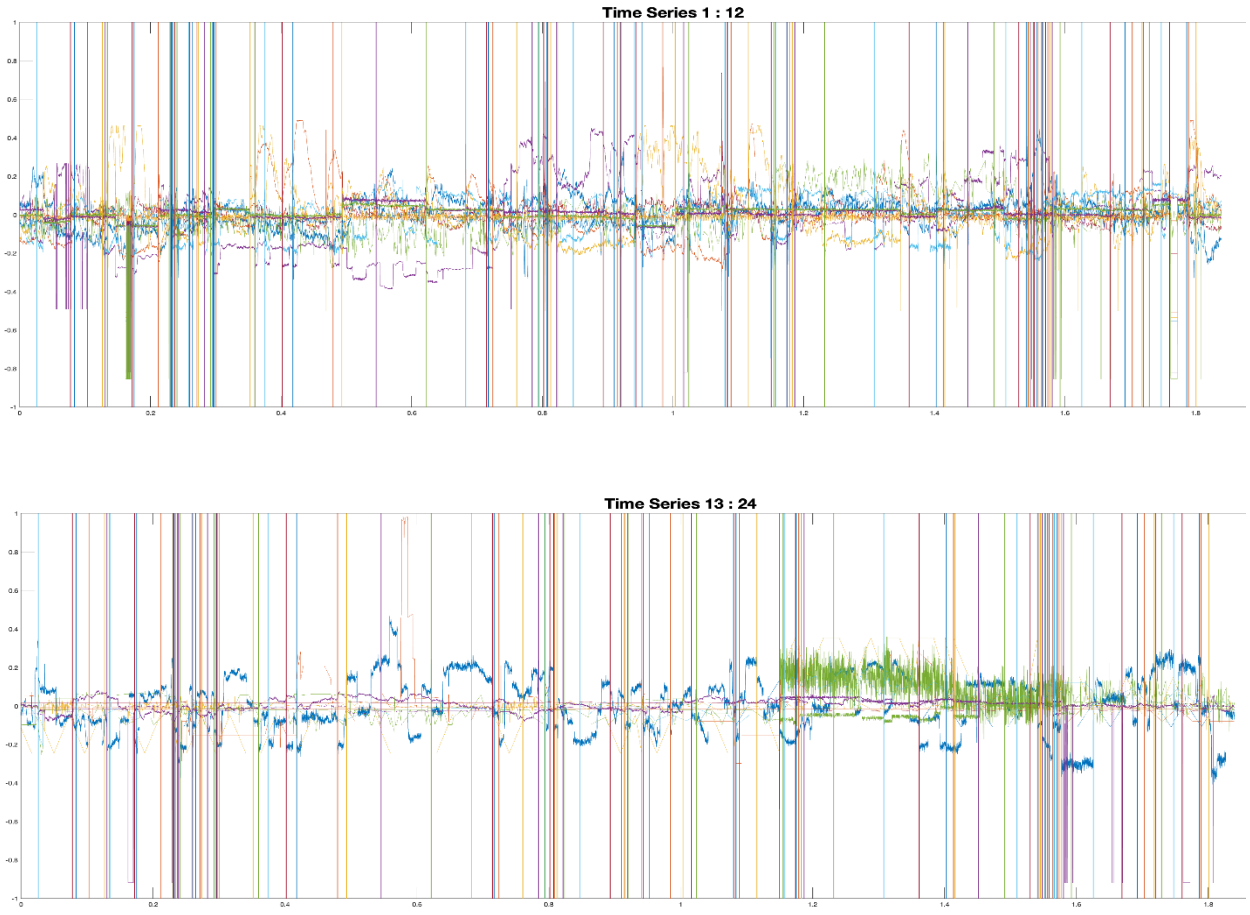


Figure 2.3: Visualization of raw sensor data from feature 1-24

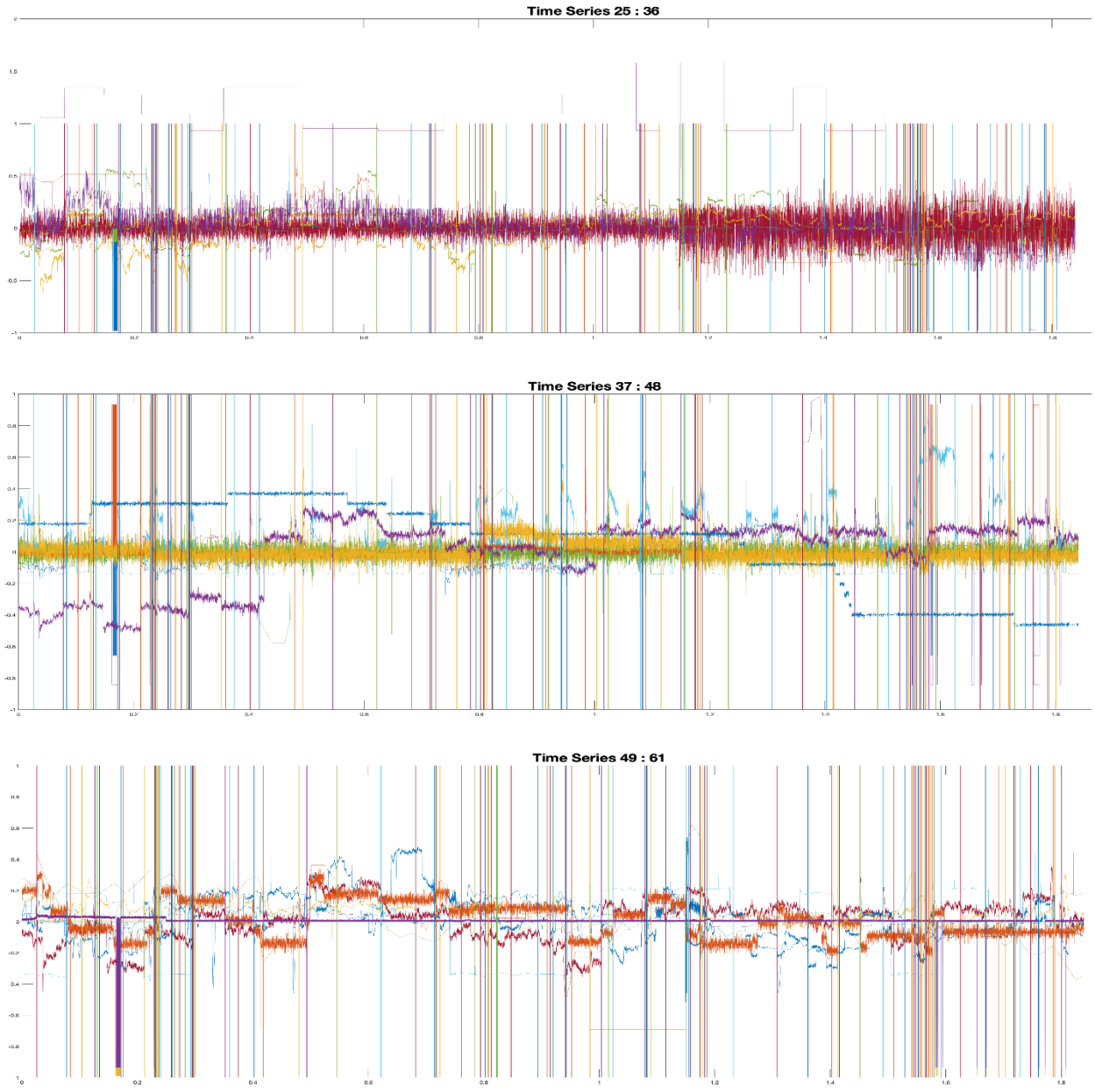


Figure 2.4*:* Visualization of raw sensor data from feature 25-61

We can check for any of the sixty-one features are good predictors for predicting y or response variable by plotting a correlation matrix. As we can see there is not even one x-variable feature which is having a good or decent amount of correlation with y-variable.

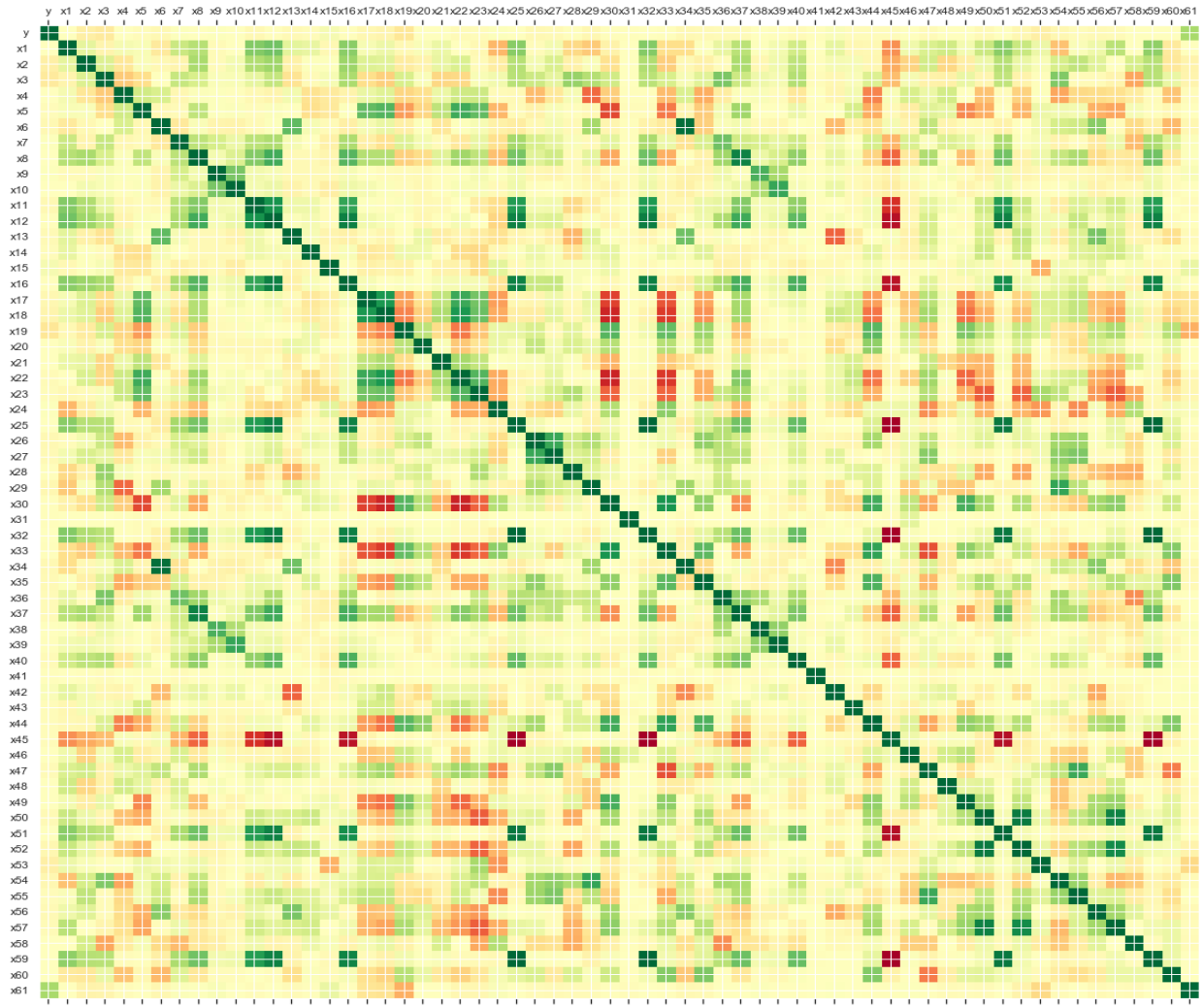
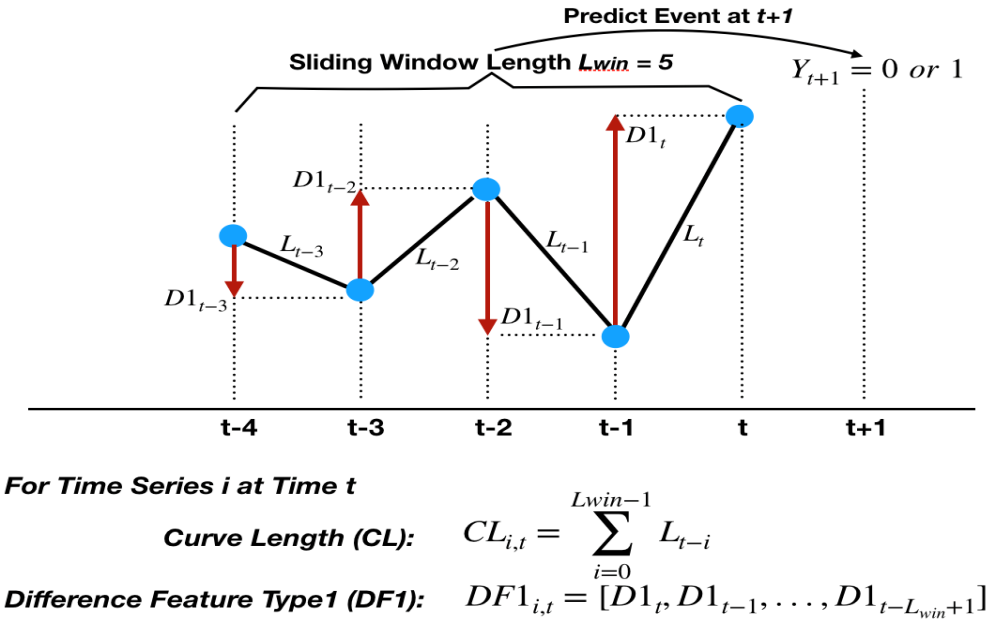


Figure 2.5: Heatmap of correlation of X variables with y

2.5 Feature Extraction

We are implementing a sliding window to compute differences for the signal or sensor reading of each feature. This will be repeated for all the sixty-one features/sensor-readings. The main use of the sliding window is to cap how early we want to predict the rare event, so for an instance, we want to predict 10 minutes early we set the length of the **Lwin** = 5 and calculate the differences. Here we are using two different functions to capture the pattern i.e. difference function-1 captures pattern from signal values continuously as per the length of the sliding window.



For Time Series i at Time t

Curve Length (CL): $CL_{i,t} = \sum_{i=0}^{Lwin-1} L_{t-i}$

Difference Feature Type1 (DF1): $DF1_{i,t} = [D1_t, D1_{t-1}, \ldots, D1_{t-L_{win}+1}]$

Figure 2.6: Difference function type-1

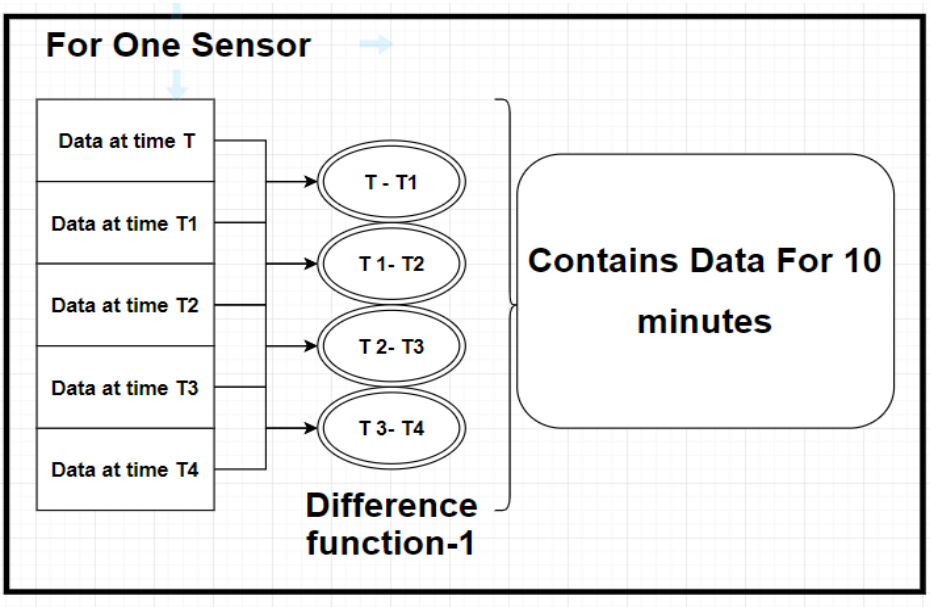


Figure 2.7: Difference function type-1 flow chart

More features are extracted using difference function type-2, we take the signal reading at position t and difference it with other signal readings in our sliding window.

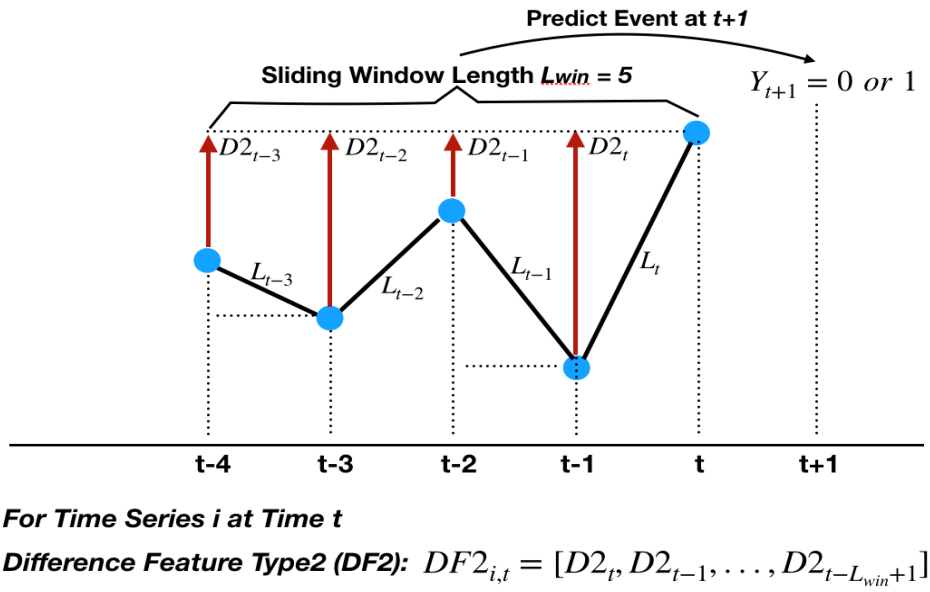


Figure 2.8: Difference function type-2

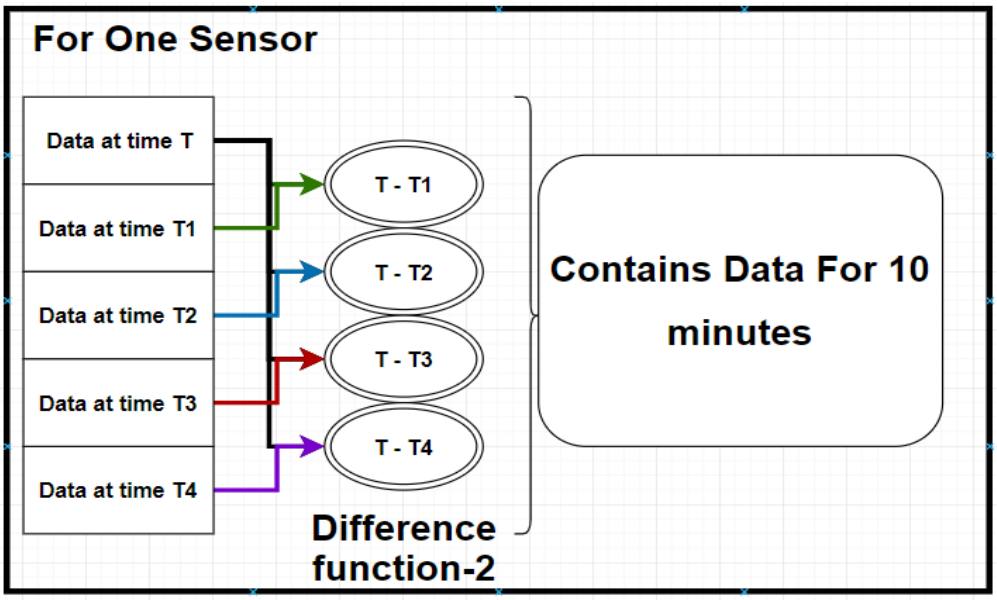


Figure 2.9: Difference function type-2 flow chart

Once we get both our difference functions, we arrange the data in following form fi,j = [CLi,t , DF1i,j, DF2i,j]. Here, CLi,t is the length of the curve of the sliding window. This process of

extracting the difference functions was done by using some simple python coding for-looping through each feature.

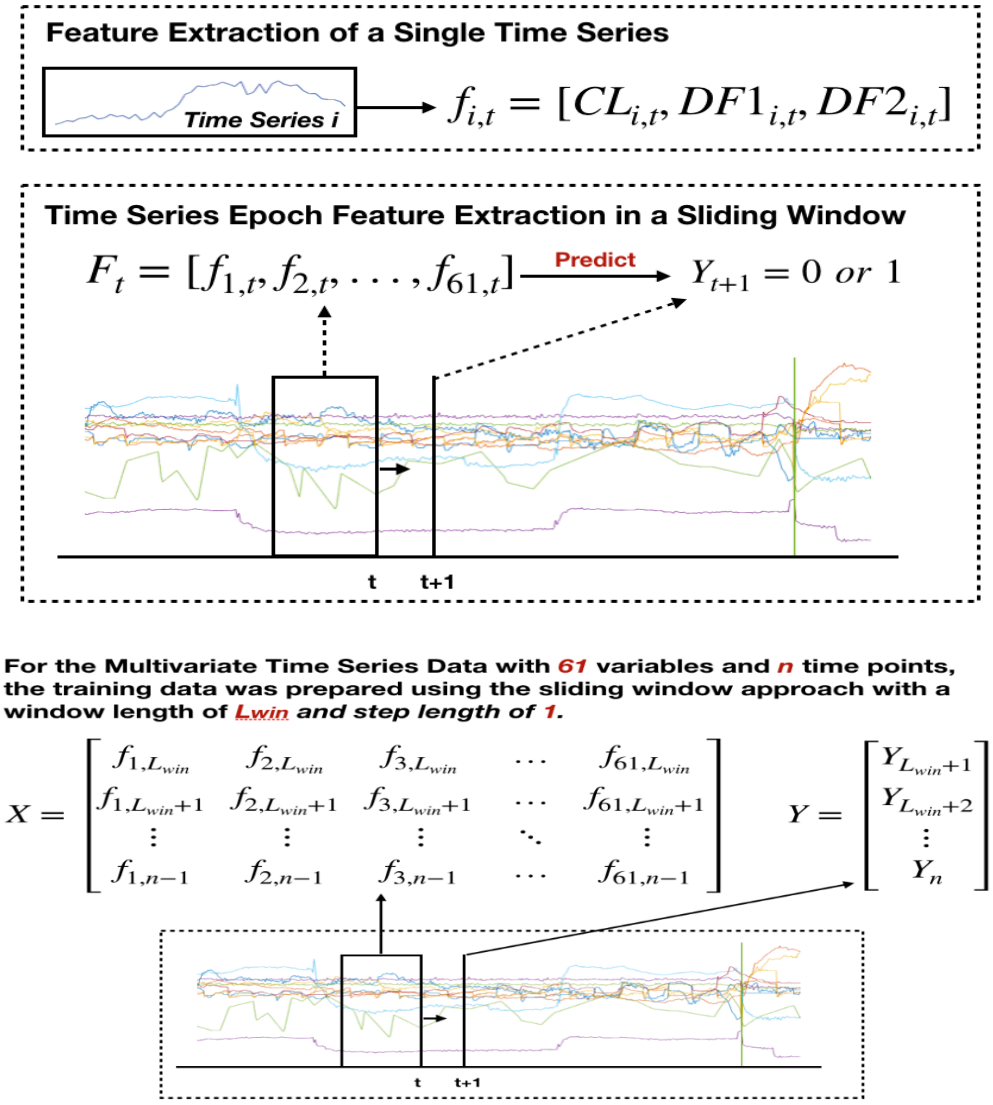


Figure 2.10: Feature extraction equation

2.6 Data Preparation for Training

We have implemented K-fold cross-validation method, this method divides the data equally into 'k' equal splits (folds). Training of k-fold cross-validation is done on all the splits and then estimates the testing set based on learning from all five splits.

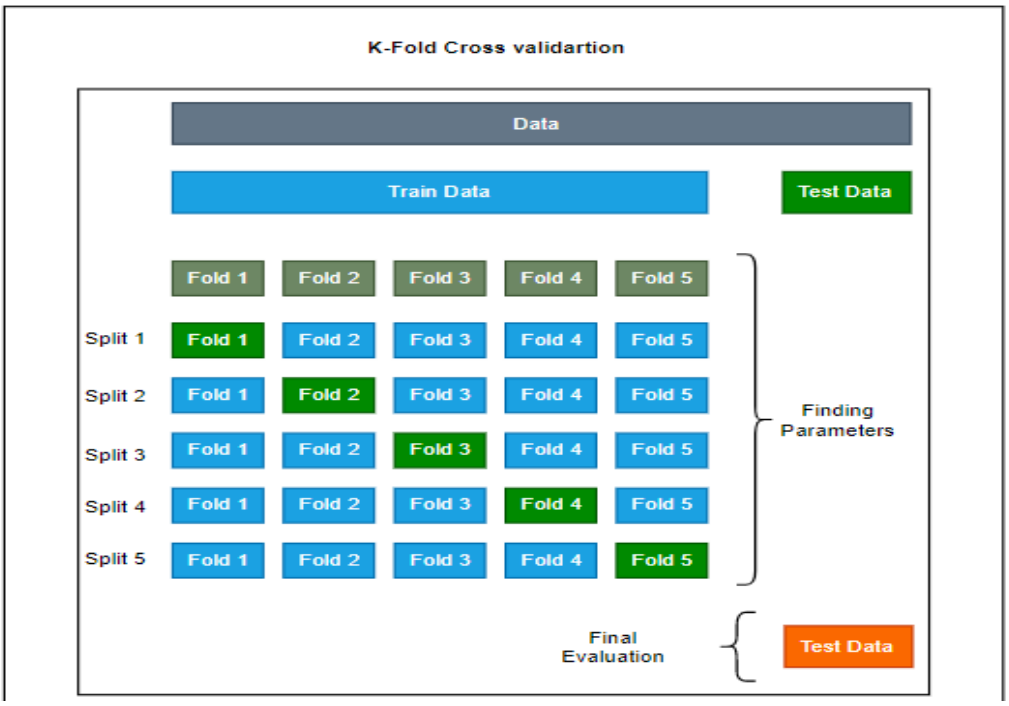


Figure 2.11: Model Training Using N-fold Cross-Validation

Next, we have implemented the oversampling method as we have only very observations of rare-event values we are applying the oversampling method. The oversampling method will only consider the minority class and creates dummy observations of minority class clusters. To be precise it creates observations by calculating the distance between halfway of first observation and next observation. We can also specify how many nearest neighbors to use for this oversampling process.

Furthermore, to get the most relevant features we have used MRMR feature selection for reducing dimensionality and work with the most relevant features that can help to predict rare events.

If *Lwin* = 4, each time has one CL feature, 3 type 1 difference features, and 3 type 2 difference features. For each time series epoch with 61 variables, the total number of features is (1+3+3) x 61 = 427.

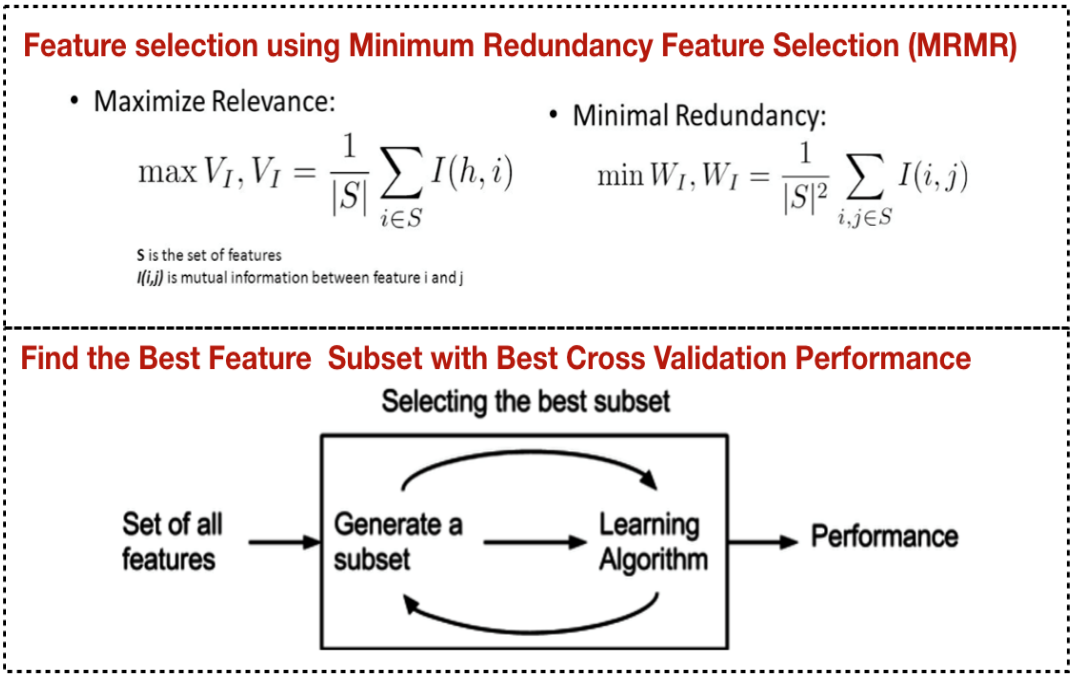


Figure 2.12*:* Feature selection

2.7 Classification Models

Machine learning mainly consists of predictive methods such as classification analysis, regression analysis, and clustering methods. In-depth machine learning has supervised learning methods and unsupervised learning methods. In general classification and regression analysis come under supervised learning methods. Clustering and dimensionality reduction models come under the unsupervised learning method. The main reason why we are implementing classification models depend on our data type, if our response variable has data type categorical, we use classification models and if our response variable has a continuous variable data type, we implement regression analysis. As describe in the problem statement we have a categorical response variable so we are implementing the following classification models.

- Deep Neural Networks.
- Random Forest.
- XGBoost.
- Ensemble Learner.

2.7.1 Deep Neural Networks

Neural Networks are build based on biological neural networks and attempt to allow computers to learn in a similar manner to humans. Neural networks are widely used on pattern recognition, time series prediction, and signal processing. A deep-neural network has multiple layers each having n number of neurons.

Our model architecture is as follows, the input layer, with 5 hidden layers each consisting of 128 neurons and an output layer with 2 neurons. The activation function used here is ‘Relu’ and ‘Softmax’. A few of the important parameters here are the number of hidden layers, the number of neurons to use in each layer, the number of epochs to use and which activation function to use. The neural networks train continuously based on the concept of gradient descent, during the training process we have to continue the loss value. Loss function tells use if the model is learning as we increase the parameters such as the number of layers, epochs to train and number of neurons once the loss function value starts to drop it means that the model is saturated and it's not learning anything new from that point of time. By monitoring the loss function, we tune our parameters number of layers, number of epochs and number of neurons. Some of the parameters which also can influence the model training well are learning rate, optimizer, batch size to input at once.

Initially while building the model we have used 32, 64, 128, 256, 512 neurons found that 128 yields the best performance. We have also played with the number of layers 3, 5, 10, 20, 50 out of

which 5 hidden layers performed the best. For epoch sizes we have implemented 5, 10, 15, 20, 30 and 50 found that the loss function stops decreasing after 6 epochs.

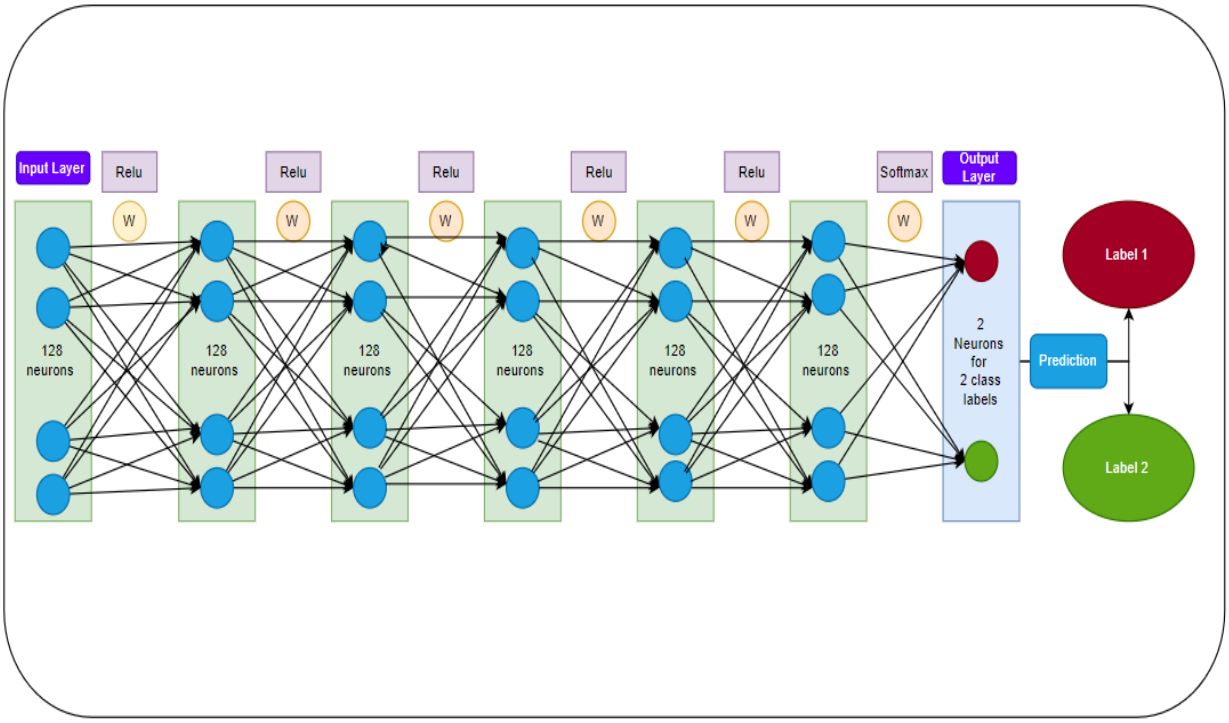


Figure 2.13: Deep-Learning model architecture

We have also implemented other classification models such as random-forest and xgboost just to get check our deep learning model is out-performing some of the baseline algorithms.

2.7.2 Random-forest classifier

Random forest classifier is easy and fast to use which provides great results most of the time without hyperparameter tuning. Random forest is a supervised learning algorithm can be used for both classification and regression problems. Random forest classifier is an ensemble version of many decision trees. The random forest includes additional randomness to the model as we

increase the number of decision trees. It uses the outputs classified by each decision tree and takes a majority vote to predict/classify the class label. Generally, more trees in the forest the more robust is the classifiers fewer chances of overfitting the model. One of the other important functions of the random forest algorithm is it can be used for feature engineering to find the most important features out of available features from training data.

Parameters tuned were number of decision trees 250, number of leaf nodes 250, class weight balanced, criterion used gini.

2.7.3 XGBoost Classifier

Xgboost is also known as extreme gradient boosting. Generally, gradient boosting classifiers take a high GPU, but xgboost classifier is designed to perform efficiently and faster. Xgboost has won many competitions on Kaggle previously, that's one of the reasons people are using it more now a days. It works on decision tree ensemble principle but keeps on decaying the loss and trains on week classifiers in the ensemble.

Parameters tuning: Number of decision trees: 200, number of leaf nodes: 200.

2.7.4 Ensemble Learner Majority-vote Classifier

We are using our outputs from both the random-forest classifier and xgboost classifier, also adding a basic neural network algorithm from sklearn to get a benchmark result on deep learning classifiers.

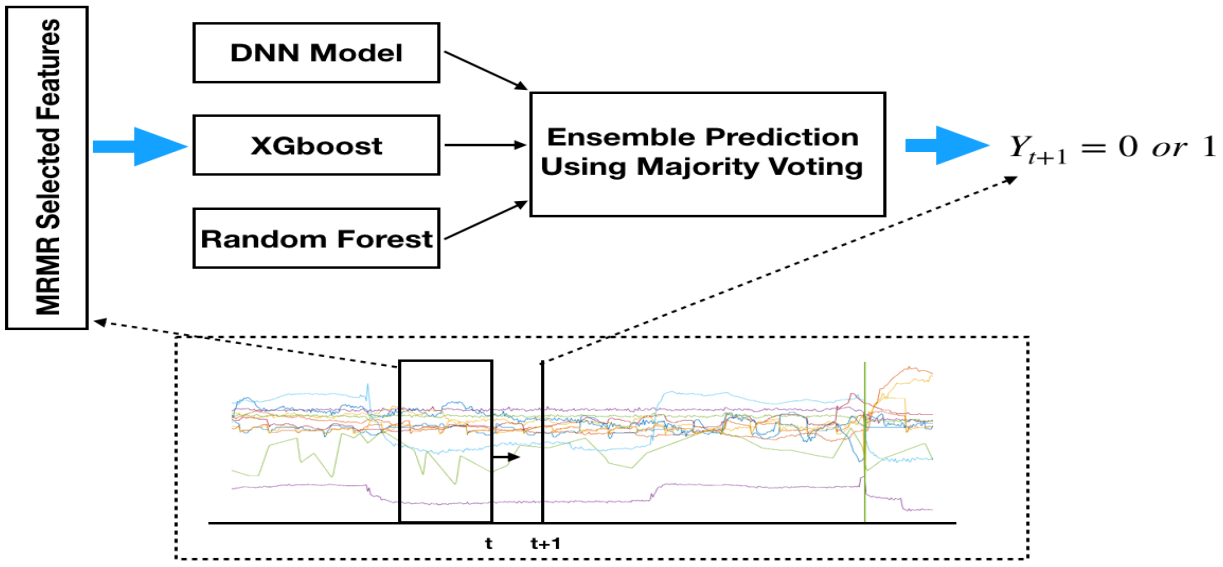


Figure 2. 14: Majority vote classifier

2.8 Results

2.8.1 Evaluation metrics for results analysis

Confusion Matrix:

A confusion matrix allows us to visualize the performance of the model. The rows in the confusion matrix represent predicted class labels by the model or classification algorithm, similarly the columns in the confusion matrix represent original class labels in the test dataset. By plotting the confusion matrix, we can calculate metrics like accuracy, precision, recall score & F-1 score.

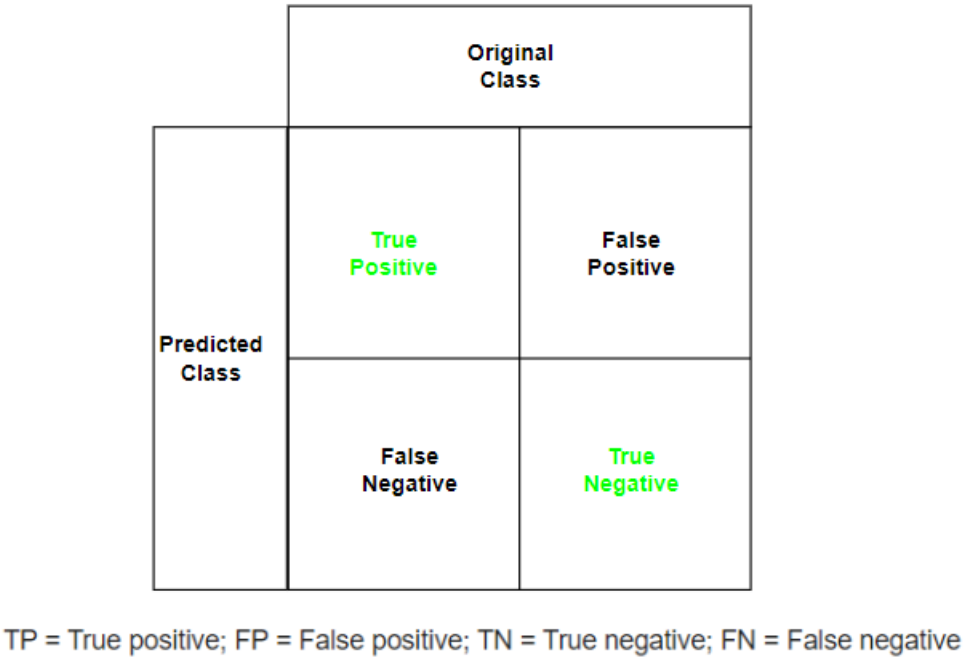


Figure 2.15*:* Confusion matrix sample

Precision:

It is also known as the ratio of true positive, precision is the ability of our model not to label an instance positive that is actually negative. Precision also captures variations in the results generated higher the precision value closer are predictions were to the actual values. This can be calculated by the formula:

Precision = **(TP)/(TP+FP).**

Recall Score:

Recall score gives us the percentage of all instances that were actually positive, what percent was classified correctly. From the recall score, we can see the percentage of false alarms occurred.

Recall = **(TP)/(TP+FN).**

F1 Score:

The F1 score is a weighted harmonic mean of precision and recall.

F1-Score = **2 (Precision * Recall) / (Precision + Recall).**

2.8.2 Results & Discussion:

From table I, table II & table III we can see that our deep-learning model outperformed all the traditional methods. As we can see all the traditional methods were able to detect patterns only for majority class perfectly but they lacked in detecting the minority class which caused the overall precision, recall & f1-score are low for all traditional machine learning algorithms.

Table I: Results Precision scores

Classifier	Precision for both classes	Precision for class no failure	Precision for class rare event
Deep-learning Neural-network	0.75	0.99	0.51
Random Forest	0.53	0.81	0.25
Xgboost	0.62	0.99	0.25
Ensemble Learner	0.7	0.99	0.4

Table II: Results Recall scores.

Classifier	Recall score for both classes	Recall score for class no failure	Recall score for class rare event
Deep-learning Neural-network	0.75	0.99	0.51
Random Forest	0.54	0.9	0.18
Xgboost	0.58	0.99	0.18
Ensemble Learner	0.56	0.99	0.13

Table III: Results F-1 scores.

Classifier	F1 Score for both classes	F1 Socre for class no failure	F1 score for class rare event
Deep-learning Neural-network	0.75	0.99	0.51
Random Forest	0.53	0.85	0.21
Xgboost	0.61	0.99	0.23
Ensemble Learner	0.62	0.99	0.25

2.9 Conclusion & Future research

Based on the results, the Deep neural network model was best in the early prediction of the rare event. Performed better when compared to baseline algorithms of the machine learning model. More parameter optimization in sliding window monitoring, such as using a longer sliding window. More investigation on feature engineering and feature selection to rare event detection.

Chapter 3

Brain EEG-data, Classification of good and bad memory encoding

3.1 Introduction to the problem statement

A typical EEG data collection process starts when an operator shows a list of words to a person and then introduces him to distraction in this case, he would the asked to solve a math problem or watch a video. Then the testing face will start by asking him if he came across the following words and the operator grades him manually. The EEG data has been collected from a basic EEG headset which contains 64 channels. Each electrode/channel collects data while the person is recollecting data during a session. We are going to grab the data 1800 milliseconds prior and 200 milliseconds after he/she responded to capture the signal pattern. This can be used to differentiate good memory encoding and bad memory encoding. Note encoding refers to how the person's EEG signal pattern is while he was memorizing a word [18].

3.2 Objective of the study

Our main goal for this case study is to grab the signal data during a word is show and use it to teach our machine learning algorithms the difference between a good encoding pattern and a bad encoding pattern. The EEG brain signal data has to be preprocessed and by using a machine learning algorithm we have to predict if EEG signals collected from a person, we have to predict him/her will memorize the words show to him/her. The EEG signals contain a lot of noise, so for this data preprocessing and making meaningful insights from data is important.

3.3 Data Collection

Data has been collected with the help of the Stereo EEG Implantation strategy. The EEG headset consists of 64 channels/electrodes to record data.

Stages in data recording:

• Encoding - Memorizing stage.

• Distraction – Introduction to simple math problems.

• Decoding - Testing stage.

The patient will be shown a list of words to recall. In our data, we have 300 un-repeated words.

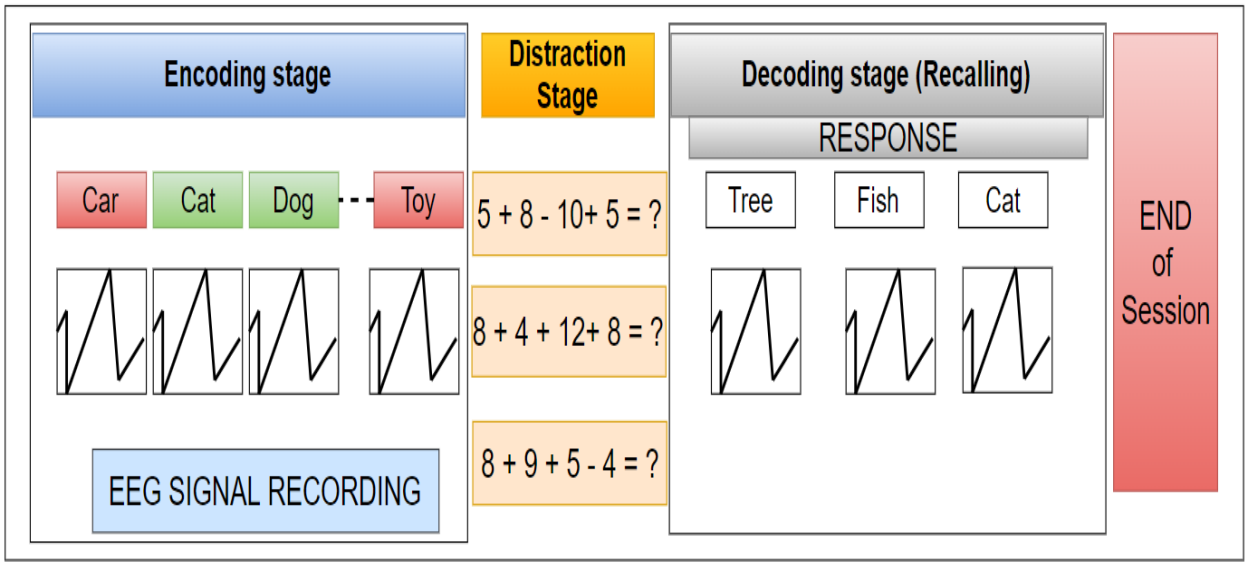


Figure 3.1: Data collection process

3.4 EEG Signal Extraction

The data was collected and stored in an EEG-events mat-lab-file, we have to extract the signal data for each word/epoch and follow our preprocessing and prediction.

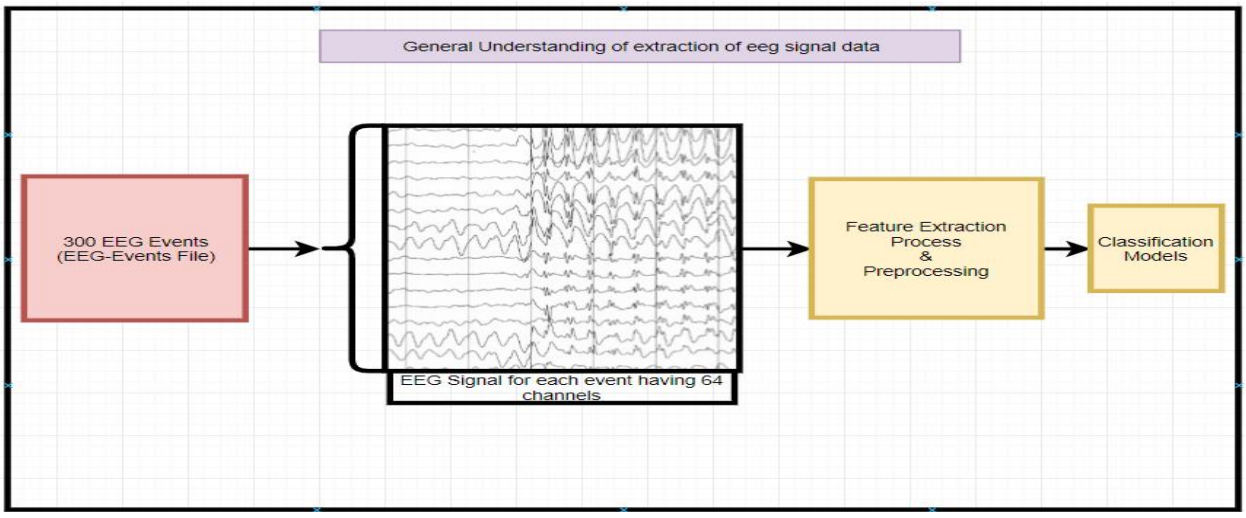


Figure 3.2: General description of data extraction and future processes

We have 300 words shown to a person and asked to memorize them, as we know our headset consist of 64 good channels, we are extracting about 1800 milliseconds of the data recorded as EEG signals for each channel. As shown in figure 3.3. Each epoch will generate a data of (360, 64) array of EEG signal values corresponding to that word/epoch. Here 360 signifies we are extracting data for 1800 milliseconds and the EEG headset will generate a signal value for every 5 milliseconds. So, for a period of 1800 milliseconds, we will have 360 readings from 64 channels.

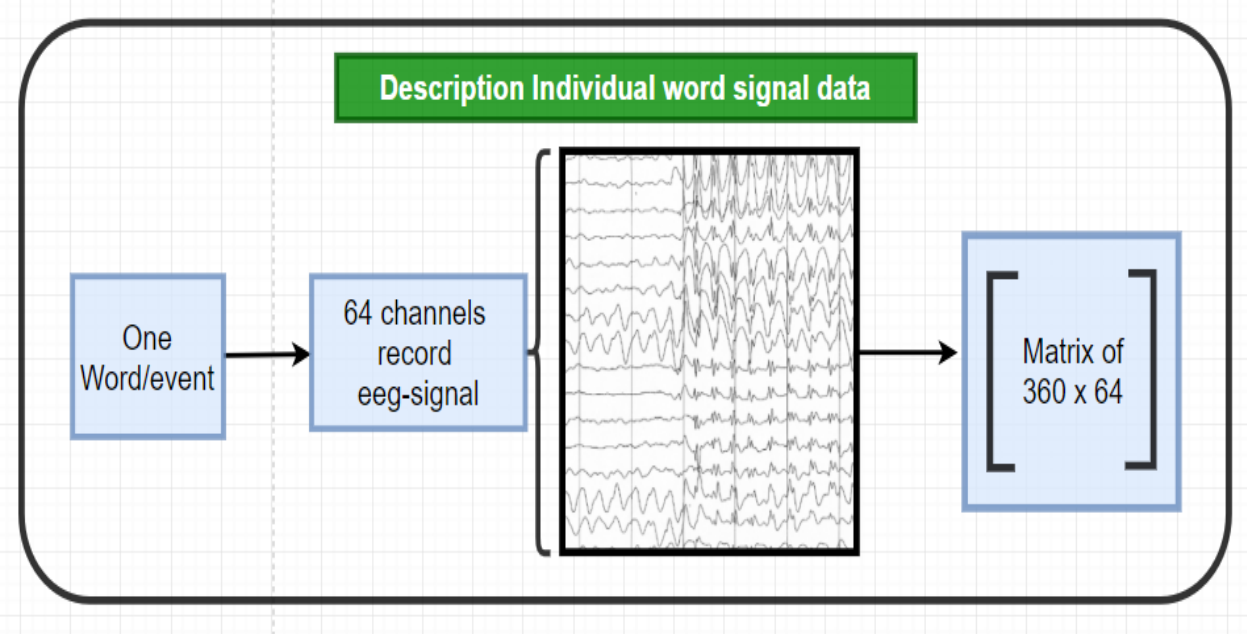


Figure 3.3: Individual eeg signal extraction.

The signal data has been extracted for a length of 1800 milliseconds, 200 milliseconds prior to memorizing and 1600 milliseconds during memorizing. For 1800 milliseconds we have taken data for every 5 milliseconds. Class variables are if the person recalled the word or not recalled. Out 300 words the person recalled 116 words and not-recalled 184 words. The reason behind taking 200 milliseconds prior to memorizing is to give a buffer on so as to capture a complete memorization pattern in the EEG signal. This process is shown in figure 3.4.

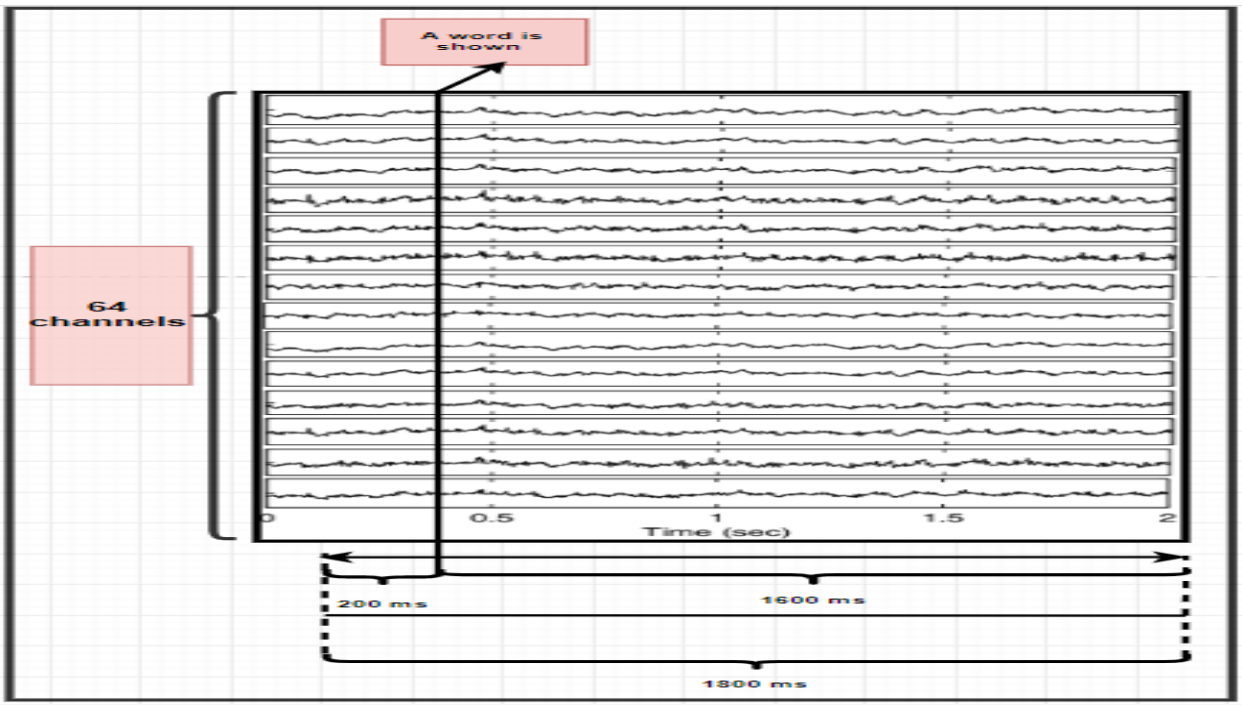


Figure 3.4: Signal extraction process

In figure 3.5, we can see how our data for 300 epochs will be looking.

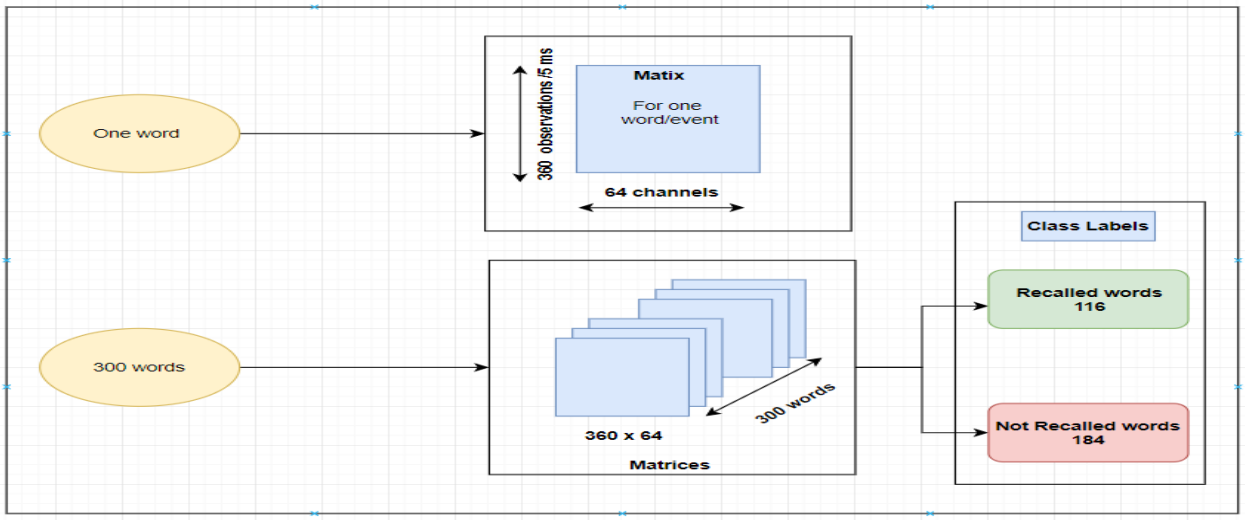


Figure 3.5: Data extraction for all 300 words/epochs

Once we have extracted the EEG-signals for all the words in EEG-file, we are going to implement feature extraction, prepare data for model and classify.

3.5 Feature Extraction

Our main goal for implementing feature extraction is to transform the EEG-signal data and help the classifiers for finding useful patterns in our data that can predict accurately.

For every epoch/word we will preprocess our original data through the following:

• Linear-correlation matrix.

• Mutual Information matrix.

• Band-power frequencies.

• General Statistics.

3.5.1 Linear-correlation & Mutual-Information

The linear-correlation matrix signifies the linear relationship between our features and labels and mutual information gives us the non-linear relationship between features and labels. So, for a 360 x 64 matrix, we will get a matrix of 64 x 64 for each linear relationship and mutual information. Keeping in mind that the matrices are symmetrical, to reduce dimensionality to a small extent we are merging one half of the linear relationship matrix to one half of the mutual information matrix. As shown in figure 3.6. This will be done individually for each epoch/word.

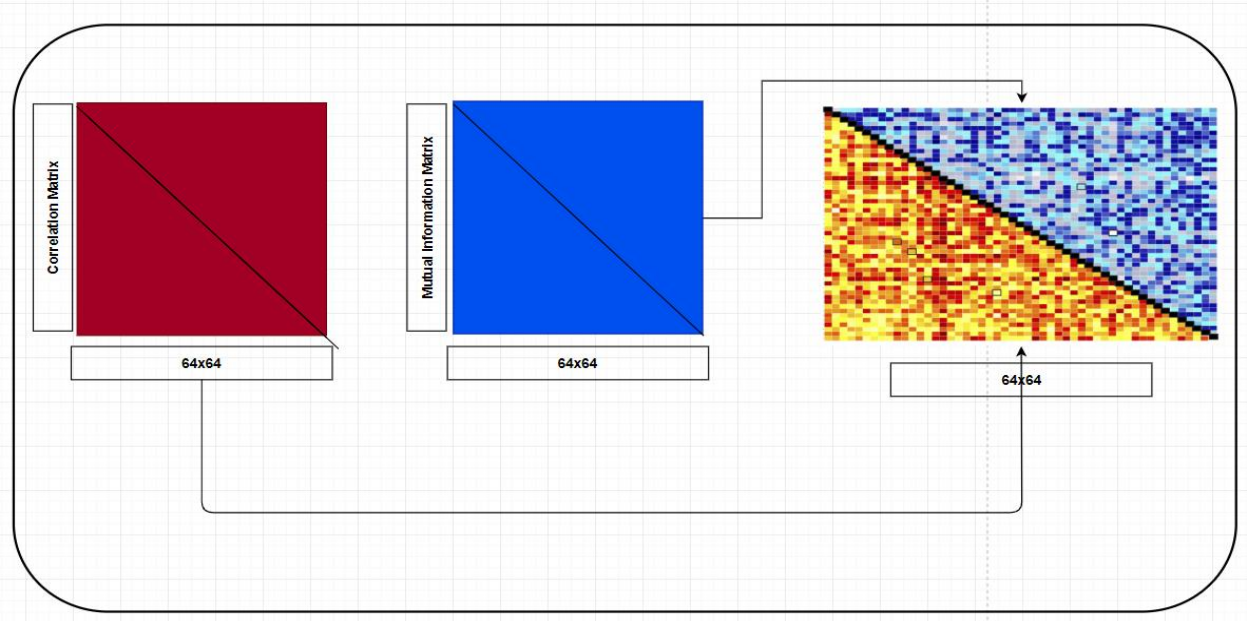


Figure 3.6: Combining linear correlation matrix and mutual information matrix.

3.5.2 Band-power frequencies

One of the most widely used method to analyze EEG data is to decompose the signal into functionally distinct frequency bands, such as delta (0.5–4 Hz), theta (4–8 Hz), alpha (8–12 Hz), beta (12–30 Hz), and gamma (30–100 Hz). Each signal is Fourier transformed into a frequency component. Each frequency band indicates the emotional state of the brain, for instance, delta waves signify a deep sleep state, theta waves signify a mild sleep state, alpha waves signify relaxed state, beta waves signify anxious state and gamma high anxiety or stress.

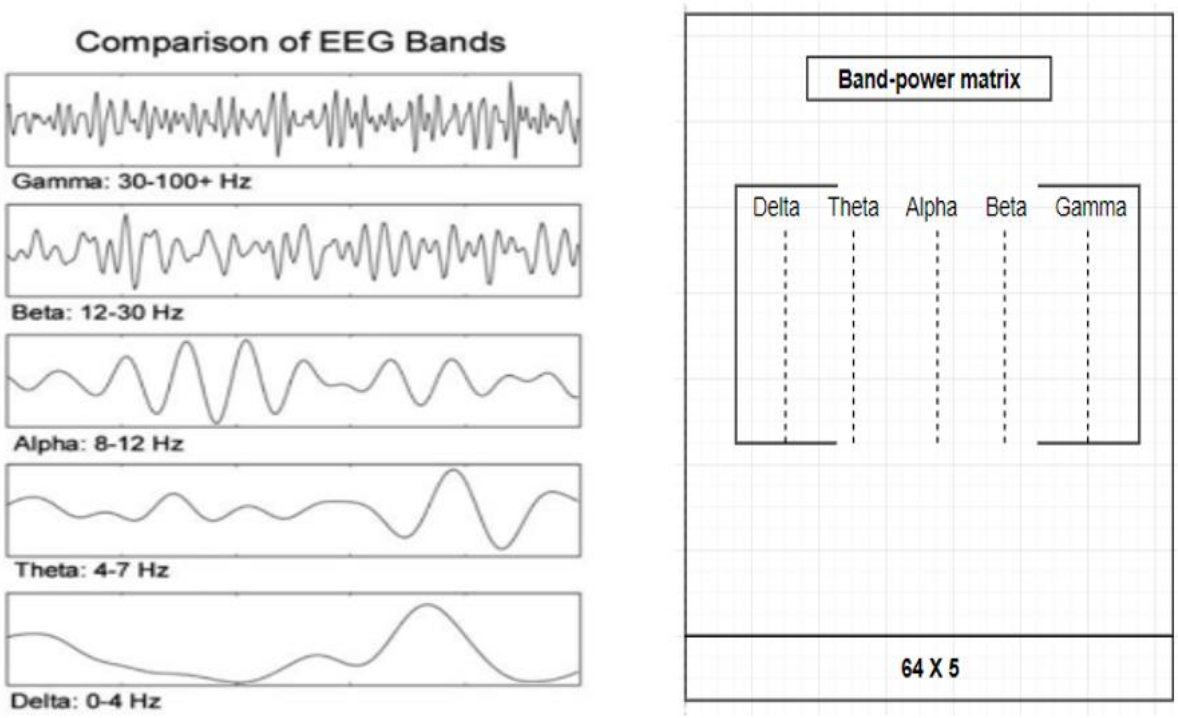


Figure 3.7: Band power general visualization

We are going to extract delta, theta, alpha, beta, gamma frequency values for all the 300 epochs with respect to 64 channels. This will yield a matrix of 64 x 5 as shown in figure 3.7.

3.5.3 General Statistics

In general statistics, we are calculating the mean, median, kurtosis & standard deviation values for 64 channels in every epoch.

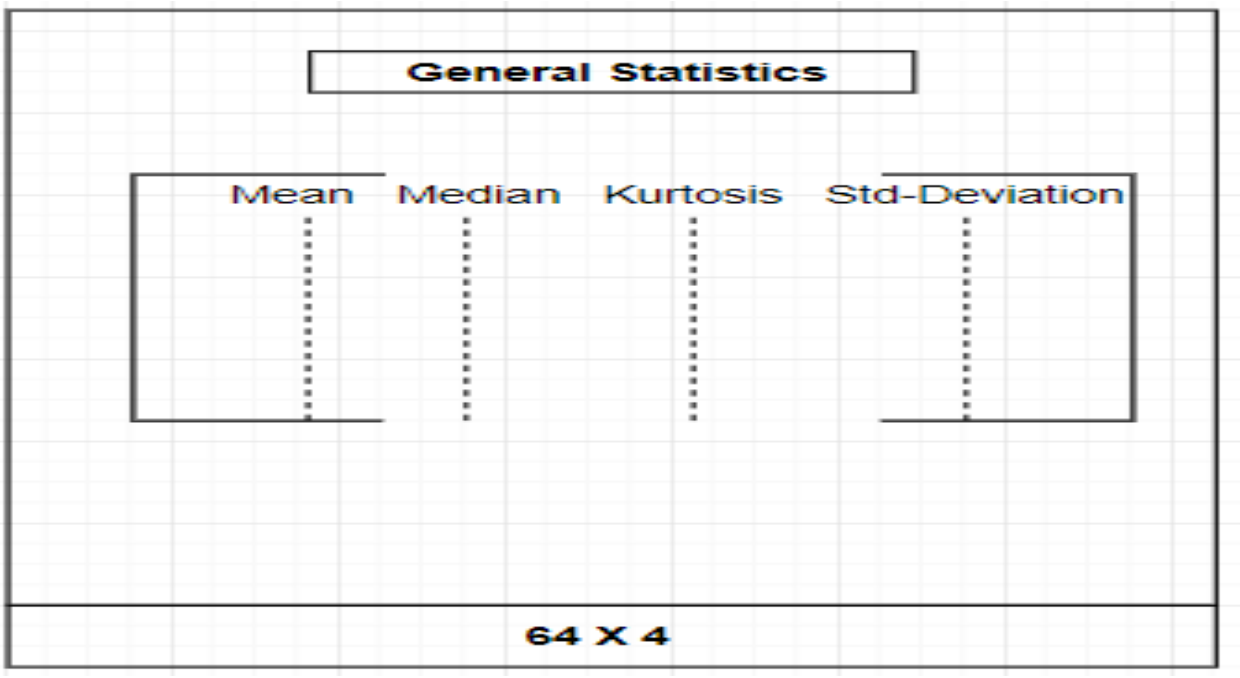


Figure 3.8: General Statistics for each epoch

By this point, we will have a matrix of 64x73 size for each epoch. So, on the total for 300 epochs, our data set size would be 300 x 64 x 73. Sample of data set in figure 3.9.

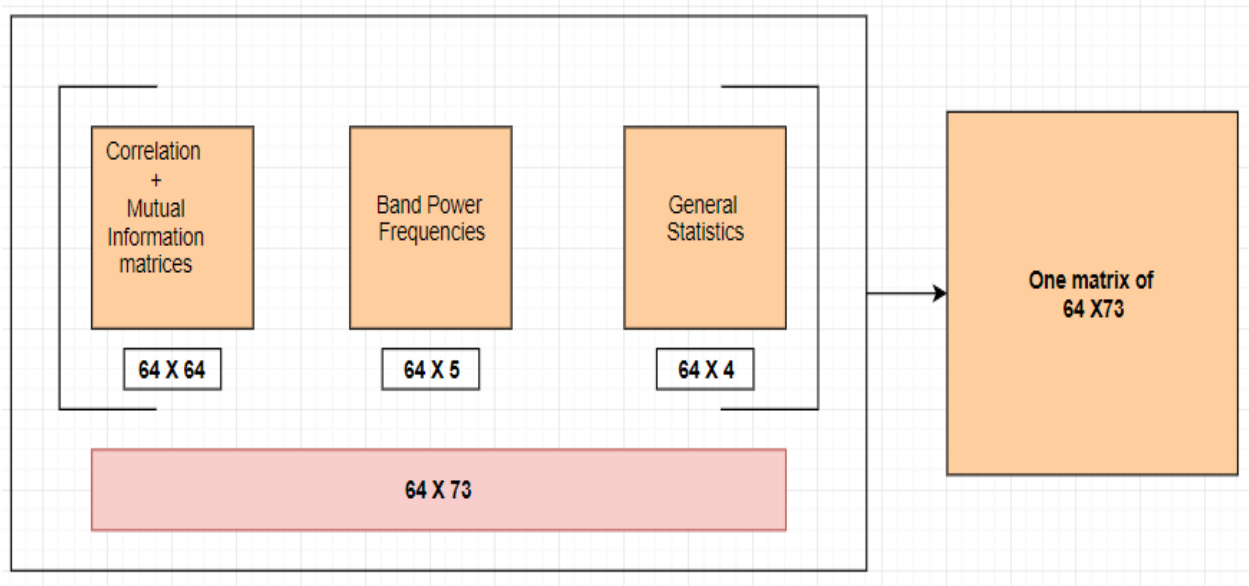


Figure 3.9: Matrix after feature extraction for one epoch

3.6 Data preparation method-1

Data preparation method-1 would be obtaining each array/matrix of 64 x 73 for all 300 words/epochs and then transforming all 300 arrays/matrices into a single array/matrix by taking each array/matrix as a row vector which will give us a new array/matrix of shape 300x4672. This method is good for traditional machine learning algorithms. A visualization of the method-1 preparation of dataset in figure 3.10.

Furthermore, we are implementing Normalization to the data using MinMaxScaler () and MRMR for getting the efficient feature to predict the class labels (Minimum Redundancy Maximum Relevance). MRMR is an approximation to maximizing the dependency between the joint distribution of the selected features and the classification variable. Selecting best size of features 10,20,50,100,200,1000 features for analysis.

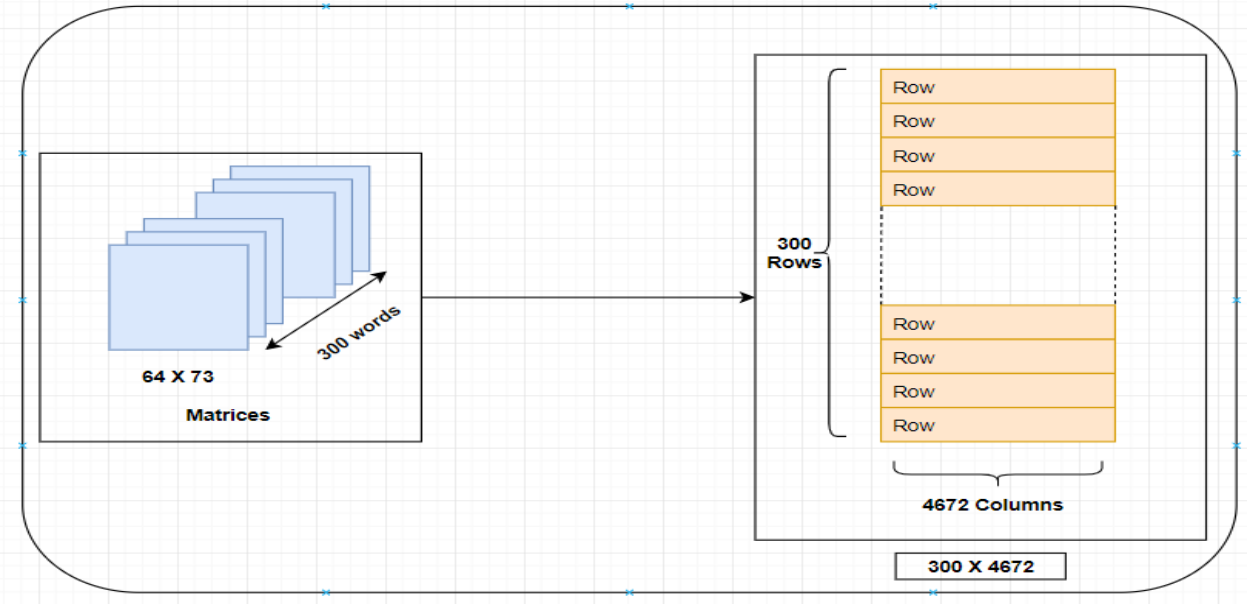


Figure 3.10: Data preparation using method-1

3.7 Models used for prediction for method-1

We have divided our training, validation & testing data sizes respectively, 80% of the data for training with 10% for validation split within and 20% as testing dataset. Classification models used are as follows:

• SVM-RBF kernel classifier.

• Random forest classifier with entropy and gini.

• Xgboost classifier.

• KNN classifier.

• Ensemble classifier with SVM-RBF, Random forest, Xgboost.

3.7.1 SVM Classifier:

Support vector machine is a supervised machine learning algorithm, SVM used a hyperplane that maximizes the margins/decision boundary between two classes. The class variables on the hyperplane are called support vectors. SVM is good for non-linearly separable data due to its kernel's. SVM's 'RBF' kernel helps to split the data into a high dimensions space due to which it can draw a hyperplane in higher-dimensional space.

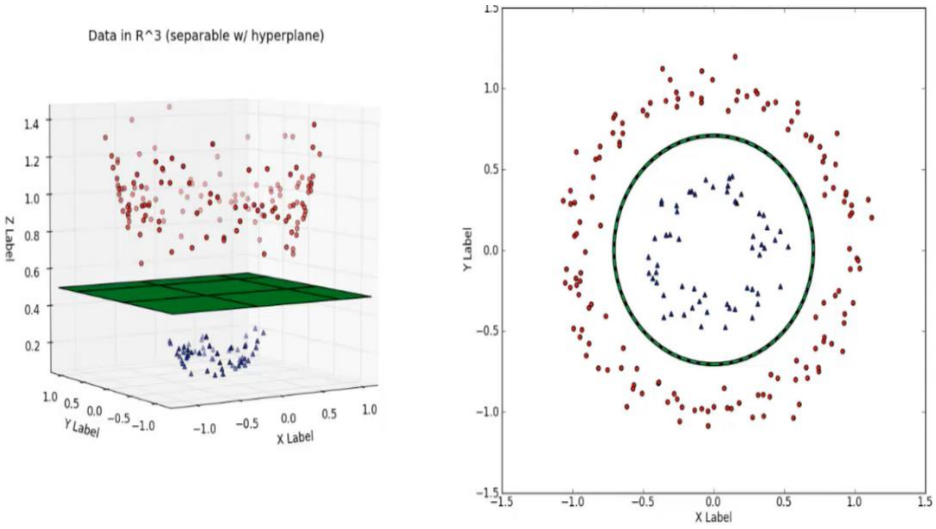


Figure 3.11: SVM hyperplane visualization

The SVM classifier parameter plays a very important role, so we have to implement grid-search to find the optimal values of C and gamma. C signifies decision function margin and gamma value signifies the complexity of the model. Lower value smoother model.

3.7.2 KNN-Classifier:

K-nearest neighbor algorithm calculates the distance between a point we want to predict and all the points in the training data. Prediction is done by majority voting of ‘k’ nearest points labels. So, if we choose our ‘k’ value to be 3, the algorithm looks at 3 closest neighbors/points labels and predicts its class label.

KNN parameters: K value signifies how many neighbors to talk vote from. Distance function we can choose which distance function to use for example Euclidean, Manhattan, etc. We have used K = 5 neighbors, Euclidean distance as a distance function to predict our results.

3.8 Data Preparation method-2

This method is focused on implementing a deep learning algorithm, we are going to transform individual matrix for each word into an image-file and going to run them through a CNN (convolution neural network). The main idea behind it is to convert the signal data for good or bad encoding into an image then use CNN, this will the computers to detect/distinguish if it was a good memory or a bad memory. If this works well the doctors can use the assistance of a deep-learning model to classify them quickly with ease.

Our main dataset already has 300 x 64 x 73 sized data, we will input each 64 x 73 matrix as an image (after transforming them into image) and teach our deep learning algorithm to classify if it's a good or bad class. Some of the difficulties we might come across are we only have 300 images in total, as per my knowledge and previous research's usually the CNN model take very high number of images to train on, as its problem in health-care we cannot ask the doctors to provide us more images as it might associate many factors such as cost, data privacy, patients time, equipment costs, etc. Sample images can be seen in figure 3.12.

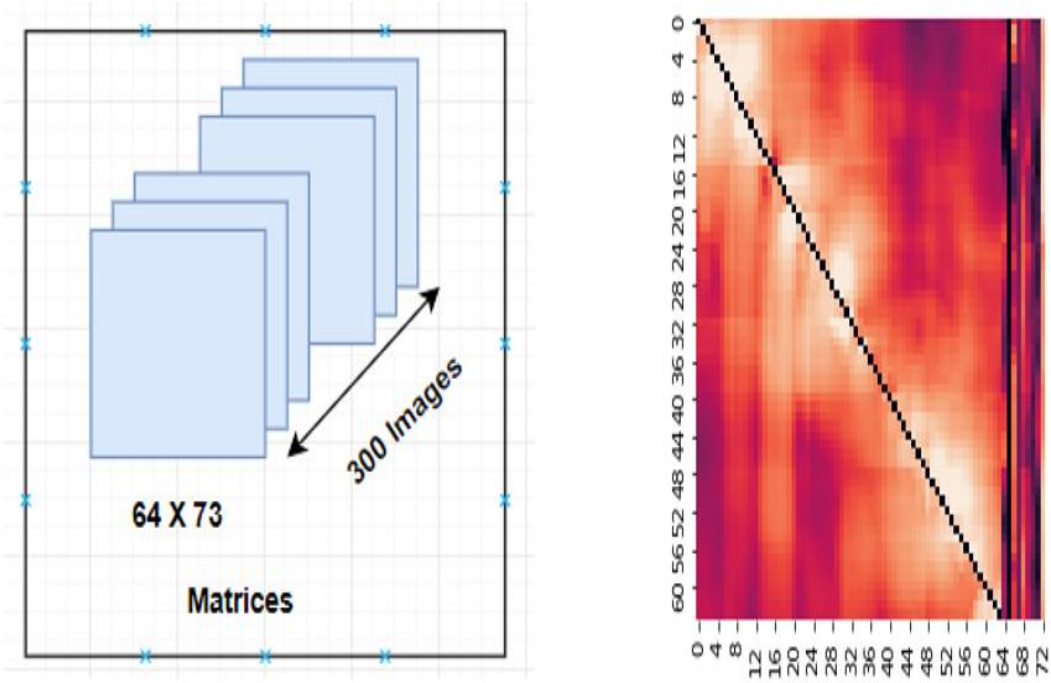


Figure 3.12: Method-2 sample of one image

3.8.1 Convolution Neural Network

A convolution neural network processes involved can be categorized as two parts feature learning and classification. Feature learning consists of three types of operations an image will go through namely convolution, activation and pooling for every layer. The classification is done on normalized outputs of the feature learning. After normalization, all the learnings are connected and then convert them into probabilities for predicting the class labels.

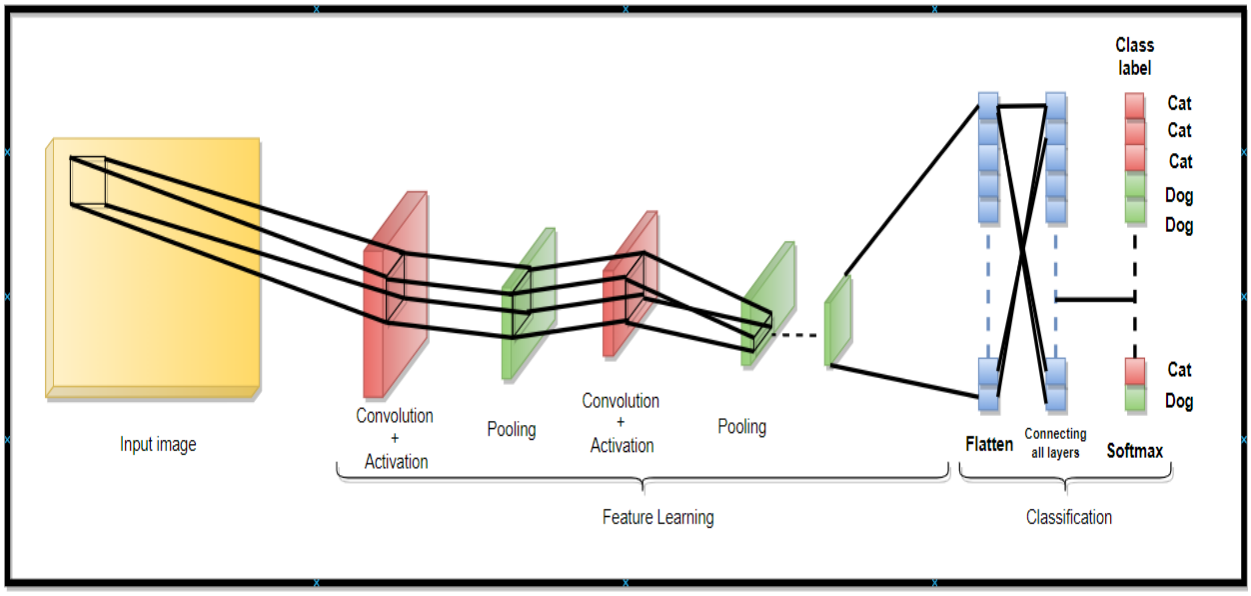


Figure 3. 13: Sample of CNN processing

3.9 Results summary & Discussion

We are evaluating our model on Precision, F1-score, recall score and accuracy. As we can see for this problem machine learning algorithm support vector machine (SVM) outperformed all other algorithms, the reason is SVM can draw a decision boundary for complex data and can recognize patterns using its hyperplane. The reason why the deep-learning algorithm could not predict better can be a lack of more training samples usually training for CNN requires thousands of images.

3.9.1 Results for best 100 features:

Table IV: Results for the top 100 features.

Classifier	Precision	Recall score	F1 score	Accuracy
Support vector machine	0.73	0.69	0.69	0.73
Random Forest	0.58	0.57	0.56	0.62
Xgboost	0.57	0.57	0.57	0.6
Ensemble Learner	0.54	0.53	0.51	0.6
KNN	0.64	0.62	0.62	0.67
Convolution Neural-network	0.5	0.5	0.5	0.5

3.9.2 Results for best 200 features:

Table V: Results for the top 200 features.

Classifier	Precision	Recall score	F1 score	
Support vector machine	0.68	0.67	0.67	0.7
Random Forest	0.56	0.55	0.55	0.6
Xgboost	0.62	0.59	0.59	0.65
Ensemble Learner	0.54	0.53	0.51	0.6
KNN	0.54	0.52	0.5	0.6
Convolution Neural-network	0.5	0.5	0.5	0.5

3.10 Conclusion & Future research

SVM was the model with the best accuracy. With an accuracy of 73% combined to classify both classes. This can be used to create a brain-computer interface and maintain a database for an individual patient to help doctors. Implementing Deep learning techniques for small datasets can be scope. Implementing Gaussian parameter tuning can help classifiers to achieve the best

parameter. For CNN generally inputting a large number of images is better, as we all know data collection in health care adds up too many complications such as cost, time, patient privacy hence we can’t demand more data from doctors so we need to improvise our algorithm to work well with fewer data samples.

REFERENCES

[1] Pat Langley, Stanford and Herbert A. Simon, Pittsburgh. "Application of Machine Learning and Rule Induction." available at http://cll.stanford.edu/~langley/papers/app.cacm.ps.

[2] AN Zeng-bo, "ZHANG Yan. The Application Study of Machine Learning[J]", Journal of Changzhi University, vol. 24, no. 2, pp. 21-24, April 2007

[3] Alpaydin, 2014, E. Alpaydin. Introduction to machine learning, MIT press, Cambridge, Massachusetts (2014), Google Scholar.

[4] Silver, D. et al. Mastering the game of go with deep neural networks and tree search. Nature 529, 484–489 (2016).

[5] Bojarski, M. et al. End to end learning for self-driving cars. Preprint at arXiv:1604.07316 (2016).

[6] S. J. Roberts, Matt Osborne, Mark Ebden, Steve Reece, Neal Gibson, and Suzanne Aigrain. 2013. Gaussian processes for time-series modelling. Philosophical transactions. Series A, Mathematical, physical, and engineering sciences 371 1984 (2013), 20110550.

[7] W. Richert, L. P. Coelho, “Building Machine Learning Systems with Python”, Packt Publishing Ltd., ISBN 978-1-78216-140-0.

[8] M. Welling, “A First Encounter with Machine Learning”

[9] M. Bowles, “Machine Learning in Python: Essential Techniques for Predictive Analytics”, John Wiley & Sons Inc., ISBN: 978-1-118- 96174-2

[10] M. van Gerven and S. Bohte, “Editorial: Artificial neural networks as models of neural information processing,” Frontiers in Computational Neuroscience, vol. 11, p. 114, 2017.

[11] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” The bulletin of mathematical biophysics, vol. 5, no. 4, pp. 115–133, Dec 1943.

[12] S. Lawrence, C. L. Giles and A. D. Back, “Face recognition: a convolutional neural-network approach,” IEEE Transactions on Neural Networks, vol. 8, no. 1, pp. 98–113, Jan 1997.

[13] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” nature, vol. 521, no. 7553, p. 436, 2015.

[14] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell, “Long-term recurrent convolutional networks for visual recognition and description,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 4, pp. 677–691, Apr. 2017. Web-link: https://doi.org/10.1109/TPAMI.2016. 2599174.

[15] X. Wu, R. He, and Z. Sun, “A lightened CNN for deep face representation,” CoRR, vol. abs/1511.02683, 2015. Web-link: http://arxiv.org/abs/1511.02683

[16] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015, pp. 3431–3440.

[17] Chitta Ranjan1, Markku Mustonen1, Kamran Paynabar, and Karim Pourak, {cranjan, markku, kpaynabar, kpourak}@processminer.com, “Rare event classification in multivariate time series” 715 Peachtree Street N.E. 100, Atlanta, GA 30308.

[18] Lega Lab: Introduction to Behavioral and EEG Analyses, UTSW.

Biographical Information

Sai Abhishek Devar was born in Hyderabad, India. He received a Bachelor's degree in Industrial and Production Engineering from Manipal Institute of Technology, Karnataka, India in 2016. He started pursuing his Masters of Science in the Industrial Engineering department in fall 2017 and is expected to graduate in fall 2019. His research interests are data science, machine learning, statistics, and python programming.