CHAPTER 6

OPTIMIZATION BASED ENSEMBLE FEATURE SELECTION AND ENSEMBLE DEEP LEARNING CLASSIFIER

6.1 ENSEMBLE CLASSIFIER AND THEIR TYPES

Ensemble classifier is a means of developing different base classifiers using which a fresh classifier can be obtained whose performances considerably superior than any individual classifier. The primary concept of ensemble strategy is to integrate a group of models, each one of which finds a solution for the same actual task, with the aim of getting an improved unified global model, that yields high accuracy and trustworthy predictions or decisions achievable with the usage of an individual model (Dong et al. 2020). The principle behind the development of a predictive model through the integration of several models has been examined for some time lately. The dissimilarity between the individual classifier and ensemble classifier is depicted in figure 6.1.

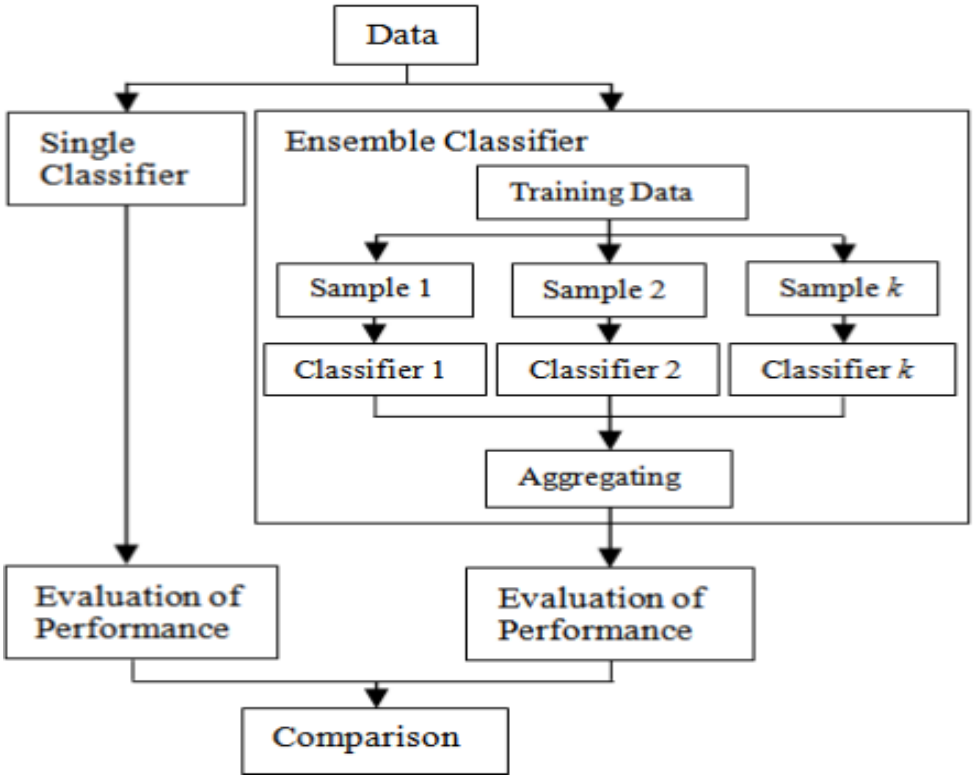

Figure 6.1. Ensemble Classifier vs Single Classifier

In ensemble learning, rather than making use of an individual classification algorithm, a group of algorithms are taken into consideration and the ultimate output is produced through the unified output of every classifier. Rather, the primary aim of an ensemble classifier is to make the best use of the strengths of different classifiers and merge their outputs in such a way that there is an improvement in the quality of the output finally. Every single classifier present in an ensemble system is called as a base classifier. The chief problems faced with ensemble classifiers include (1) selection of the base classifiers and (2) merging the output of the base classifiers (Kadkhodaei et al. 2020). One important cue to design an effective ensemble is to make sure that the base classifiers are adequately different (Thomas et al. 2018, Cavalcanti et al. 2016). Two classifiers are different when no correlation exists between their outputs. Several techniques exist for developing an ensemble system. If homogeneous ensembles are considered, the base classifiers are taken from the same group and the versatility among the base classifiers is attained through their training with various samples of the training dataset.

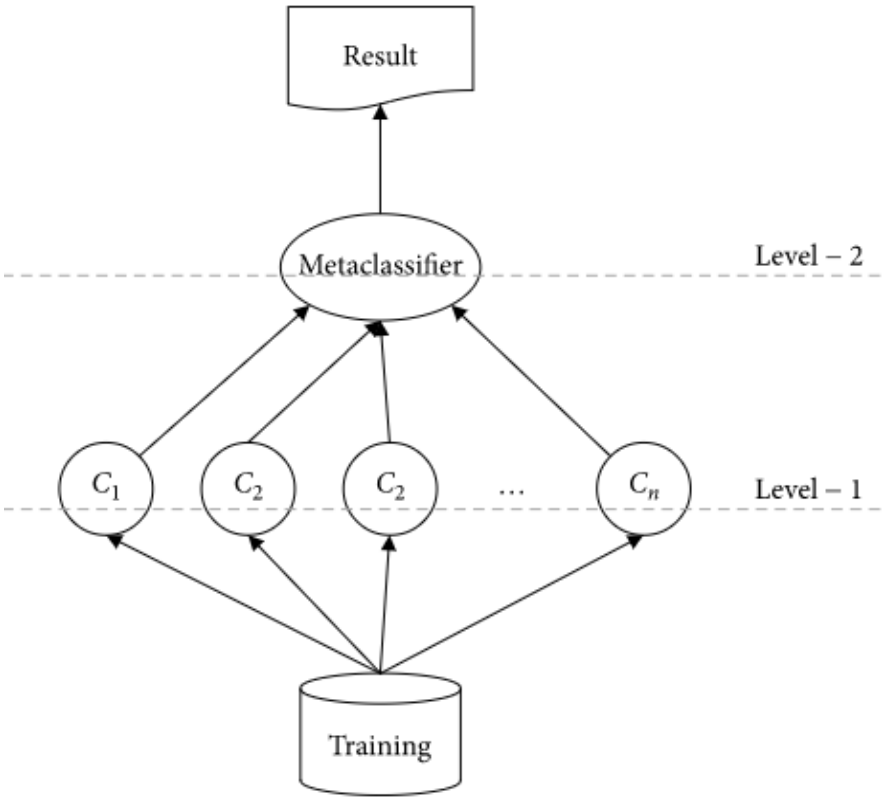

Figure 6.2. The Stack Generalization Approach

On the contrary, in heterogeneous ensembles, various classification algorithms are taken into consideration for the base classifiers; and hence, versatility is gained when a variety of algorithms are used (van Rijn et al. 2018). In Bagging, every base classifier gets trained using a bootstrapped copy of the training dataset and the ultimate decision is formed through the application of a majority voting over the decision made by every base classifier. It has to be noticed that other combination techniques are also published in the literature, namely weighted voting and plurality voting (Sultana & Islam 2020). On the other side, the Boosting builds the base classifiers iteratively, and each one balancing the setbacks of its predecessors. The developed base classifiers are later combined with the weighted voting technique (Kadkhodaei et al. 2020).

Stack Generalization (in short, Stacking) is one of the highly effective heterogeneous ensembles. It can be seen from Figure 6.2 that a two-layer schema is used to train the model. In the first layer, the training of the base classifiers is performed with various classification algorithms on the actual dataset. Once the base classifiers in the first layer are trained, their output for the same actual dataset is utilized in the form of a new dataset for training the met classifier in the second layer. It implies that the output of every classifier in the first level is treated as the inputs to the meta-classifier in the second level. It is noticeable that choosing what kind of the base classifiers is to be used in the first level is a significant challenge in the Stacking, since it could have an effect on the prediction performance of the Stacking. Like it is stated earlier, an optimal subset of the base classifiers is that showing versatility and accuracy. The prediction performance of the Stacking completely relies on the accuracy and versatility of the base classifiers in the first layer (van Rijn et al 2016).

6.2 CONTRACTIVE AUTO-ENCODER (CAE)

CAE are essentially unsupervised DLT that assist NN in encoding unknown training data. Auto encoders are often utilized to learn a representation, or encoding, for a large amount of unknown data as the initial step toward decreasing dimensionality. The contractive auto encoder reduces the sensitivity of encoding to minor changes in its training dataset. This is accomplished by adding a regularizer, or penalty term, to any cost or objective function that the algorithm attempts to reduce. As a result, the learnt representation sensitivity to the

training input is reduced. This regularizer must adapt to the Frobenius norm of the Jacobi matrix for the encoder activation series with regard to inputs.

The Frobenius norm is alternatively known as Euclidean norm. Jacobian matrices are defined as matrices having all 1^{st}order partial derivatives of vector valued functions and hence are squares were both matrices and their determinants are called as Jacobians. The principle behind demising the autoencoder is only to improve the reliability of the encoder with regard to the slight variations in the training data that is identical to the inspiration behind Contractive Autoencoder. But, some difference exists,

- CAE promotes effectiveness of representation f(x), while DAE promotes reliability of reconstruction, due to which just a partial increase in the robustness of representation is observed.
- DAE improves its consistency by random training of the model for the reconstruction, while in CAE; the consistency of the first derivative of Jacobian matrix is improved.

6.3 SPARSE AUTO-ENCODER (SAE)

Sparse Autoencoder (SAE) is a special kind of AE, which depends on regularization. For the regularization of AE, sparsity constraints, a penalty term for the loss function is also included. A huge number of (instead of fewer) hidden units can be present compared to the inputs, but just a low number of the hidden units are permitted to activate simultaneously. This helps the model in learning improved representations of input compared to the traditional AE. It is worth noticing that when SAE have the capability of reducing the data dimension, they can also find the intriguing forms in the input data. On the whole, the first SAE layer is utilized for extracting the feature vector from the input, which is later utilized in the form of the input for the upcoming SAE layer. This process goes on till the training is finished in an unsupervised fashion. Later, back propagation algorithm (BP) is used for reducing the cost function and improves the weights by taking the labeled training set into consideration to attain supervised classification.

6.4 PROPOSED METHODOLOGY

Dimensionality reduction is achieved with the help of Kernel based principle component analysis , Optimization based ensemble feature selection by Fuzzy monarch butterfly optimization algorithm, LFCSA, and AFA, Ensemble deep learning classifiers by FCBi-LSTM, CAE, and SAE, and at last, the results of all the classifiers are evaluated in terms of performance metrics such as F-Measure, MCC, Accuracy, and Error. Figure 6.3 illustrates the overall flow of the proposed approach.

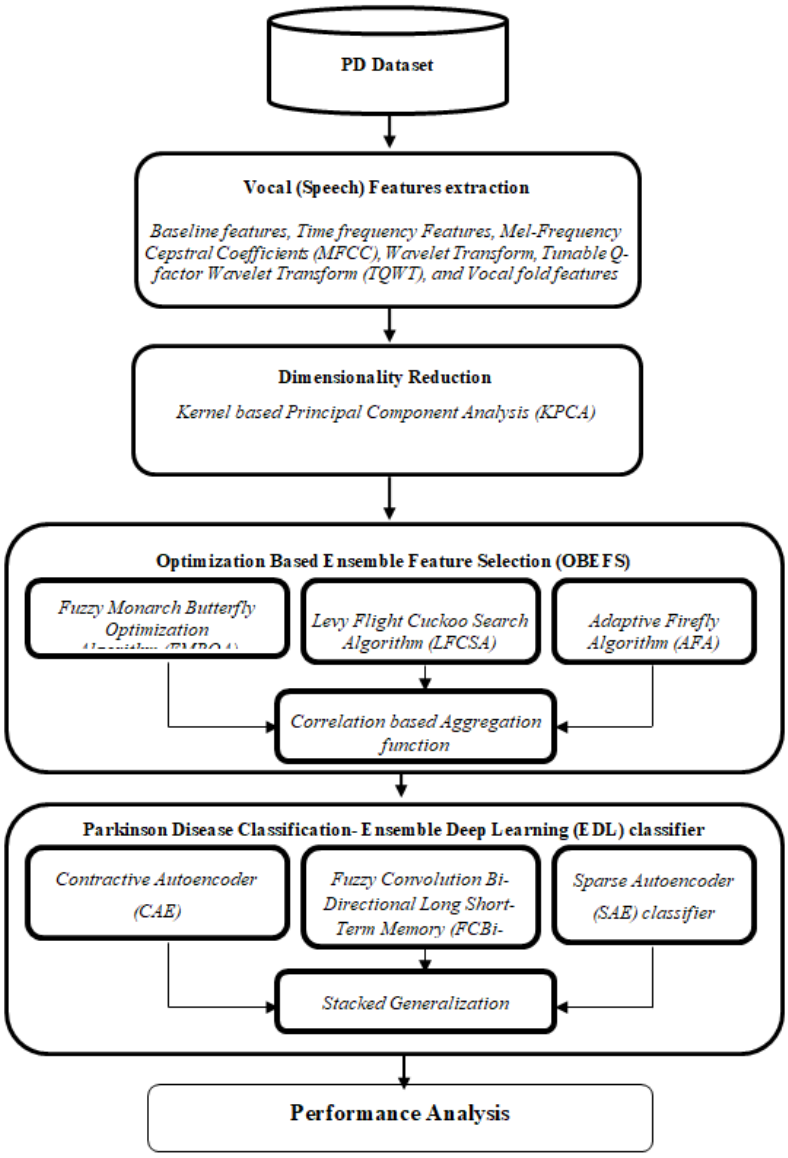

Figure 6.3. Overall Flow of Proposed Scheme

6.4.1 Optimization Based Ensemble Feature Selection (OBEFS)

OBEFS utilises several FS techniques and merges their normalized outputs to quantitative ensemble significance. OBEFS approach is achieved by collecting the feature rankings that every single feature selector including FMBOA, LFCSA, and AFA provides, into an ultimate joint ranking through correlation.

6.4.1.1.Fuzzy Monarch Butterfly Optimization Algorithm

FMBOA is introduced for the features sets selection. FS of each sample is achieved with the help of FMBOA, and the amount of effect exerted by features on the existence of PD is assessed. FMBOA chooses the features for samples in accordance with the level of the effects on presence. MBO depend on the migratory nature (Feng et al. 2018). FMBOA have been used for getting reasonable results in classifications if they are used with no changes. This suggests that they make an effort to strike a balance between local and global searches. In order to improve information for systems (Feng et al., 2018) where search operations and data distribution are algorithmically conducted utilizing operators specifically for butterfly and migrations adaptations, butterflies associate and share information locally inside the swarm.

6.4.1.2.Levy Flight Cuckoo Search Algorithm

CSA (cuckoo search algorithm) depend on cuckoo species that take advantage of appropriate hosts for optimizing the choice of features from datasets for progressing their progeny. The purpose of CSA is to reduce the possibility of losing eggs (or superfluous traits) to other species while maintaining parental responsibility for improving their progeny. The ultimate qualities are decided by dropping the eggs (features) in multiple nests. LFCSA begins with the populations made up of N host nests (Mesa et al. 2018). A cuckoo egg (feature), assume, is selected randomly during every iterations' t, and the novel features solutions f_i^{t+1} are formed. The Lévy flights are a type of random walk in which the steps have certain probability distributions and are depicted in regard to step-lengths. The steps must have random and isotropic directions. When calculating replacement rates stochastically and optimizing for better results, probabilities prb_a are useful. All of the present solutions are rated according to their fitness (accuracy) over the course of iterations, with the best features

(optimal solutions) obtained thus far being kept as feature vectors fbest. The process is repeated till the predefined termination criterion is attained.

6.4.1.3. Adaptive Firefly Algorithm

The optimal behaviour of firefly flashing forms the building block for FA (Wang et al. 2017). The steps of Fire flies have to located at a distance to the maximum extent possible from the optimal answers. To select the features from datasets in an optimal manner, fireflies are used and there exists a balance between global and local searches. Consequently, the movements of the firefly have to also take the historical data and data current location into consideration. This investigation takes into account Firefly historical data, which contains the best results from the preceding two iterations. Fireflies use an iterative process to determine their next actions depending on the spaces between their present fitness values and population optimal fitness values. Iterations may result in changes to the steps, and firefly steps are also subject to modification.

6.4.1.4.Correlation function

Correlation coefficients matrix is computed for the attributes that are chosen in the outcome of every single technique applying the Equation (6.1) below,

$$CorrelationCoefficient = \frac{N\sum xy - (\sum x)(\sum y)}{\sqrt{\left[N\sum x^2 - (\sum x)^2\right]\left[N\sum y^2 - (\sum y)^2\right]}} \tag{6.1}$$

Where, x and y stands for the attributes values considered and N refers to the overall numbers of instance. The classification process uses the characteristic set that the correlations-based ensembles characteristic selectors select as the input.

6.4.2 PD classification using EDL classifier

In this research, the classification of PD is achieved with the help of an EDL classifier. Ensemble learning can substantially improve the generalization strength of learning system. Since its introduction, EDL has been incredibly successful in PD classifications. It is based on the ML technique that utilizes multiple classifiers, like FCBi-LSTM, CAE, and SAE, to develop an ensemble learner utilizing stacked generalization to facilitate better generalization

of learning devices. The EDL classifier is successful in learning several base classifiers and collates their results applying particular conditions. The rule utilized for aggregating the outputs specifies the outstanding performance of an ensemble.

6.4.2.1.Fuzzy Convolution Bi-Directional Long Short-Term Memory (FCBi-LSTM)

The FCBi-LSTM is used to categorize objects based on features selected from PD datasets. Pooling and convolution layers define these networks. After applying convolutions, they pool the data such that the outputs can be utilized as inputs for further convolutional layers. To achieve a better representation, filter-based convolution layers split the features into multiple matrices of size (Nm+1). CNN uses its pooling layers to minimize the size of the output matrices while preserving the relationships between the features.. Bi-LSTM performs better than uni-directional LSTM and preserves more structural data since it uses the outcomes of the final convolution layer as intermediary variables. The two Convolution layers process the Bi-LSTMs' final results in order to diagnose Parkinson's disease. The approach known as MFB (multi-modal factorized bilinear pooling) is applied to combine data from CNNs with the characteristics obtained from Bi-LSTM.

6.4.2.2.Sparse Auto-encoder (SAE)

The SAE classifier neurons labeled as (+1) supply the bias units to the FFNN (feed-forward NN) using cost functions. Figure 6.4 presents the model's outline. DNN uses the sparse auto encoder's bottlenecks as input vectors.

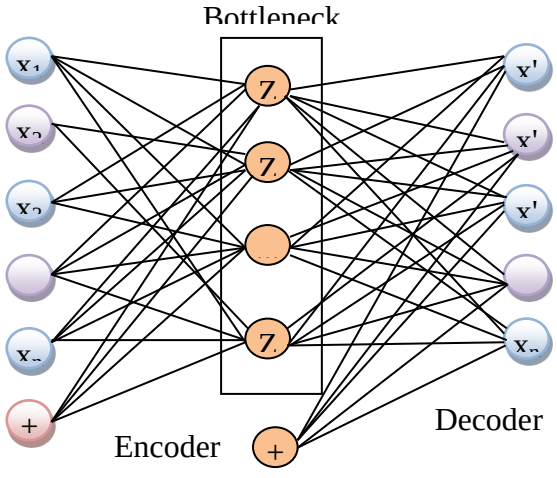

Figure 6.4. Sparse Model of SAE

SAE are introduced in which the cost functions constitute three segments. Presuming that the dataset contains N training instances $(x_1, x_2, \ldots x_n)$, where x_i refers to the i^{th} input. SAE learn reconstructing inputs x_i applying the cost functions $h_{W,b}(x_i)$ close to x_i where MSE, weight decays, and sparsity are added to the function. Equation (6.2) specifies the cost functions for the weight decays and MSE of N training samples.

$$J_{sparse}(W,b) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\left\|h_{W,b}(x^i) - x^i\right\| + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{l}\right)^2 \tag{6.2}$$

Weight decays given by the above Equation (6.2) prevents over fitting of data, as small values of λ can result in overfitting of data, whereas its bigger values can lead to underfit data. The third component of the cost functions, sparsity, prevents overfitting of the data and is helpful in activating the hidden layers of AE. It is capable of to reducing the number of hidden layer regions that are assessed. Applying Equation (6.3), where an is the activation function and incorporates the rectifier (ReLU), one can get the average active value of the hidden layer.

$$\hat{p}_j = \frac{1}{N}\sum_{i=1}^{N}\left(a_j^2(x^i)\right) \tag{6.3}$$

Sparsity is computed to get $\hat{p}_j$closestto p (sparsity parameter) and it aids in deviating from p and either activates or de-activates the neurons of the hidden layers. It can be specified with the help of Kullback-Leibler Divergence (KLD) as shown in Equation (6.4),

$$\sum_{j=1}^{s_l} KL(p\|\hat{p}_j) = \sum_{j=1}^{s_l}\left[plog\frac{p}{\hat{p}_j} + (1-p)\log\frac{1-p}{1-\hat{p}_j}\right] \tag{6.4}$$

SAE cost functions include the results of all the three components expressed in Equation (6.5),

$$J_{sparse}(W,b) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\left\|h_{W,b}(x^i) - x^i\right\| + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{l}\right)^2 + \beta\sum_{j=1}^{s_l} KL\left(p\|\hat{p}_j\right) \tag{6.5}$$

where β indicates the sparse penalties. The training of the deep NN classifier is achieved with the help of the bottleneck of the SAE in the form of inputs, and the training of the SAE is performed to reduce their cost function as it was earlier stated. The training of both SAE and the classifier were performed simultaneously, leading to an enhanced feature extraction when the selection of the classifier was idealized. The training process goes through 30 iterations, employing 8 instances for each batch. While the weight decay λ was fixed at 0.0001 and the sparse penalty term at 2, the sparsity parameter p was fixed at 0.05. In the last ten iterations, DNN is adjusted to change the classifier's parameters, and while the SAE parameters remain unchanged, the softmax cost function is decreased. The parameters are revised with the help of the Adam optimizer based on the computed gradients.

6.4.2.3. Contractive Auto-encoder (CAE)

Contractive AE modify the learned representations and they exhibit robustness towards slight variations close to the training examples [30]. It is able to achieve this by utilizing another penalty term exerted on the representation. In the reconstruction term scenario, the loss function (ℓ2) is applied. AE employs the encoder function f to convert the inputs x onto the inner representations (or codes) f(x). Then, the mapping of f(x) back to their input spaces is done using the decoder functions g. Functions for reconstruction loss The errors are penalized using l, where r(x) represents the x predictions. Reconstruction functions r are made up of the functions f and g, i.e., r(x) = g(f(x)). CAE [30] fall under normalized AE, which are used in learning the minimization of the regularized reconstruction errors and it is expressed as,

$$\mathcal{L}_{CAE} = \mathbb{E}\left[\ell\big(\mathrm{x}, \mathrm{r}(\mathrm{x})\big) + \lambda \left\|\frac{\partial \mathrm{f}(\mathrm{x})}{\partial \mathrm{x}}\right\|_{\mathrm{F}}^{2}\right] \tag{6.6}$$

Where $r(x) = g(f(x))$ and $\|\mathrm{A}\|_{\mathrm{F}}^{2}$ provides the total of the squares for each of A's elements. Both the squared loss,

$$\ell(\mathrm{x}, \mathrm{r}) = \|\mathrm{x} - \mathrm{r}\|^{2} \tag{6.7}$$

and the loss of cross-entropy,

$$\ell(\mathrm{x}, \mathrm{r}) = -\mathrm{x}\log \mathrm{r} - (1 - \mathrm{x})\log(1 - \mathrm{r}) \tag{6.8}$$

Due to the simpler mathematical approach facilitated, the examination is focused on the squared loss. One must note that the parameterizations of f and g, specifically the connected weights condition that the Equation (6.9), depends critically on the reduction of the CAE conditions. (19).

$$f(x) = \text{sigmoid}(Wx + b) \tag{6.9}$$

$$g(h) = sigmoid(W^T h + c) \tag{6.10}$$

The following regularizing term forces f (and g) to be contractive, that is, to include singular values less than 1, because of the associated weights. Greater values of λ result in steeper contractions (lower singular values) in the local directions with little to no data difference, where they have the least impact on reconstruction mistakes.

6.4.2.4. Stacked Generalization

Stacking generalization forecasts the generalized biases with respect to the particular learning sets. To create a suitable linear ensemble of the base learners, the optimal ensemble weights in the regression were found using least squares with non-negativity conditions and cross-validation data [30]. Consider the linear ensemble of f1, the predictions of the base learners. $f_2, \ldots, f_m$ expressed using Equation (6.11),

$$f_{stacking}(x) = \sum_{j=1}^{m} w_j f_j(x) \tag{6.11}$$

where w indicates the optimal weight vector that the meta learner learns.

6.5 Experimental Results

The experimental observations of the suggested EDL classifier are elaborated in this section, along with a comparison analysis of it with other methods, such as FCLSTM-CNN dependent on CNN, FCBi-LSTM, and CNN. The program used to analyze PD detections and their classification was MATLAB R2016a. The Intel(R) CoreTMi3-4160T CPU@3.10 GHz 3.09 GHz processor, 4.00 GB RAM, Windows 8.1 pro, 64-bit OS, operating system, and 1 TB hard disk are the system specifications used throughout the implementation stage. Evaluation metrics validate the classifiers' prediction performance analysis. F-Measure, MCC, error, and

accuracy are some of the evaluation measures used to assess how predictably the classifiers operate.

For this experiment analysis for sparse encoder we use hidden layer size of 10, hyperparameters like L2 Weight Regularization as 0.001, Sparsity Regularization as 4, and Sparsity Proportion as 0.05 and Decoder Transfer Function as logsig. For contractive auto-encoder the number of neurons is 50 and we calculate the jacobian matrix. The constrains and their optimal values obtained is described.

Table 6.1 Accuracy Comparison Results of EDL Classifiers with Triple Feature with KPCA+ OBEFS

Feature Extraction Combination	EDL Classifier	FCBi-LSTM	FCLSTM
TQWT+MFCC+Wavelet	97.771	96.6381	93.0470
TQWT+MFCC+Concat	99.156	98.0244	93.0854
TQWT+ Wavelet + Concat	98.378	97.3457	93.1261
MFCC + Wavelet + Concat	99.903	98.7720	95.1557

Table 6.1 describes the Accuracy Comparison Results of Classifiers with Triple Feature with KPCA+ OBEFS. In this result MFCC+Wavelet+Concat shows the higher accuracy compared to the other feature combinations. It gives 99.903% of accuracy for Ensemble Deep Learning classifier, 98.7720% of accuracy for FCBi-LSTM Classifier and 95.1557% of accuracy for FCLSTM Classifier.

Table 6.2 F-measure Comparison Results of EDL Classifiers with Triple Feature with KPCA+ OBEFS

Feature Extraction Combination	EDL Classifier	FCBi-LSTM	FCLSTM
TQWT+MFCC+Wavelet	99.449	98.3100	94.2258
TQWT+MFCC+Concat	97.722	96.5900	91.5250
TQWT+ Wavelet + Concat	98.652	97.5200	93.4200
MFCC + Wavelet + Concat	99.633	98.5010	91.6921

Table 6.2 describes the F-measure Comparison Results of Classifiers with Triple Feature with KPCA+ OBEFS. In this result MFCC+Wavelet+Concat shows the higher accuracy compared to the other feature combinations. It gives 99.633% of F-measure for Ensemble Deep Learning classifier, 98.5010% of F-measure for FCBi-LSTM Classifier and 94.2258% of F-measure in the comparison of TQWT+MFCC+Wavelet for FCLSTM Classifier.

Table 6.3 Mathews Correlation Coefficient Comparison Results of EDL Classifiers with KPCA+ OBEFS

Feature Extraction Combination	EDL Classifier	FCBi-LSTM	FCLSTM
TQWT+MFCC+Wavelet	75.432	74.300	67.6669
TQWT+MFCC+Concat	73.412	72.300	67.7060
TQWT+ Wavelet + Concat	71.013	70.300	65.4457
MFCC + Wavelet + Concat	72.431	71.400	67.2960

Table 6.3 describes the MCC Comparison Results of Classifiers with Triple Feature with KPCA+ OBEFS. In this result TQWT+MFCC+Concat shows the good MCC result compared to the other feature combinations. It gives 71.013% of MCC for Ensemble Deep Learning classifier, 70.300% of MCC for FCBi-LSTM Classifier and 65.4457% of MCC in the comparison of TQWT+MFCC+Wavelet for FCLSTM Classifier.

6.6 SUMMARY

This chapter studies about solving a multi-class classification problem with the help of EDL classifier. OBEFS and EDL classifier is proposed for PD evaluation. OBEFS is run utilizing algorithms like FMBOA, LFCSA, and AFA. Correlation function is utilized for selecting the best features from the three subset features. Next, the Ensemble Deep Learning classifier is used over the FCBi-LSTM, CAE, and SAE. The stacked generalization technique was designed with the intent of collecting the inferences of classifiers while selecting the majority class (PD) of the dataset. Classification techniques and their results are assessed in regard to F-Measure, accuracy, errors, and MCC. The proposed EDL classifier by MFCC+Wavelet+ Concat ensemble yields maximum accuracy rates of 99.9030%, F-measure rates of 99.633%, and MCC rates of 72.431%.