Feature Based Fuzzy Membership: A Novel Fuzzy SVM Method for Imbalanced Learning in Thyroid Classification

I. INDRODUCTION

Thyroid disease is a medical condition that harms the function of the thyroid gland. It keeps thyroid from making the right quantity of hormones. These disorders may lead to enlargement of the thyroid gland, causing direct symptoms such as difficulty in swallowing and neck discomfort. It is very common throughout the world and causes problems because of misfunctioning of the thyroid gland. [1] References to goiter have been seen in texts dating back to 2700 BC. It is said that the Chinese were aware of the enlarged thyroids from around 2700 BC. The Chinese, Egyptian, Indian, Greek and Byzantine medicines had huge contribution. Along with developments in diagnosis, clinical thyroidology also grew in the 20th century, Charles Mayo first used the term hyperthyroidism in 1907. To help detecting thyroid disease in the healthcare industries, supervised machine learning algorithms can be very helpful. But traditional machine learning classifiers are sensitive to imbalance dataset. Since real world data cannot be balanced or free from noise, the supervised algorithms fail to detect disease accurately. Fuzzy Support Vector Machine is a novel method of classification that incorporates fuzzy logic in SVM (Support Vector Machine) algorithm to help detect data having class imbalance problem. A new fuzzy membership method is proposed which is based on the distance from significant features threshold.

II. Support Vector Machine

SVM (Support Vector Machine) is one of the efficient supervised learning models for classification purposes. It classifies data using support vectors which are the closest data points of separate classes. It is to find the optimal hyperplane that can separate the data points in different classes in the feature space. The hyperplane tries to make the margin between the support vectors as maximum as possible.

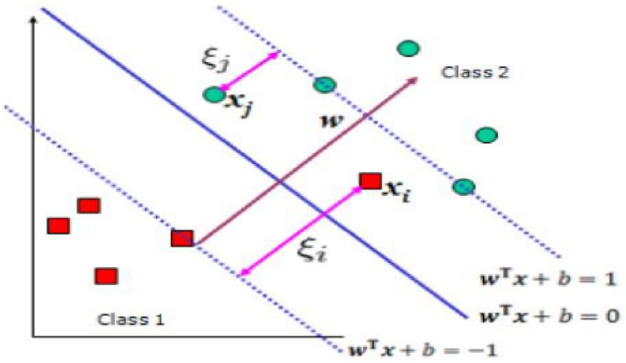

Figure 1: SVM with Slack variables misclassified data

[2] Classification function for linearly separable data:

$$f(x) = sign(w^T x + b) \tag{1}$$

[3] If the dataset is completely linearly separable, the separating hyperplane with the maximum margin can be found by solving the following maximal-margin optimization problem:

$$\begin{aligned} &\text{Min}\left(\frac{1}{2} w \cdot w\right) \\ &\text{s.t. } y_i(w \cdot \Phi(x_i) + b) \geq 1 \\ &i = 1, \ldots, l. \end{aligned} \tag{2}$$

However, in most real-world problems, the datasets are not completely linearly separable, although they are mapped into a higher dimensional feature space. The soft-margin optimization problem is formulated as follows:

$$\begin{aligned} &\text{Min}\left(\frac{1}{2} w \cdot w + C \sum_{i=1}^{l} \varepsilon_i\right) \\ &\text{s.t. } y_i(w \cdot \Phi(x_i) + b) \geq 1 - \varepsilon_i \\ &\varepsilon_i \geq 0, \qquad i = 1, \ldots, l. \end{aligned} \tag{3}$$

The slack variables εi > 0 hold for misclassified examples and the parameter C can be treated as the misclassification cost of a training data. This quadratic-optimization problem can be solved by constructing a Lagrangian representation and transforming it into the following dual problem:

$$\begin{aligned} &\text{Max } W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \Phi(x_i) \cdot \Phi(x_j) \\ &\text{s.t. } \sum^{l} y_i \alpha_i = 0, \qquad 0 \leq \alpha_i \leq C, \qquad i = 1, \ldots, l \end{aligned} \tag{4}$$

Lagrange multipliers should satisfy the following Karush–Kuhn–Tucker (KKT) conditions:

$$\begin{aligned} &\alpha_i(y_i(w \cdot \Phi(x_i) + b) - 1 + \varepsilon_i) = 0, \\ &(C - \alpha_i)\xi_i = 0, \qquad i = 1, \ldots, l. \end{aligned} \tag{5}$$

By applying a kernel function the optimization problem becomes:

$$\begin{aligned} &\text{Max} W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ &\text{s.t. } \sum_{i=1}^{l} y_i \alpha_i = 0, \qquad 0 \leq \alpha_i \leq C, \qquad i = 1, \ldots, l. \end{aligned} \tag{6}$$

Finally, the SVM decision function is given by:

$$f(x) = \text{sign}(w \cdot \Phi(x) + b) = \text{sign}\left(\sum_{i=1}^{l} \alpha_i y_i K(x_i, x) + b\right). \tag{7}$$

The existing SVM classification model works better for balanced dataset, but it shows difficulty classifying minority class in imbalanced dataset [2].

III. CLASS IMBALANCE LEARNING

Various methods have been adopted to solve class imbalance problem in recent years. One way is to balance the dataset using random under sampling, oversampling and SMOTE methods. [4] Using a geometric model, the k-Nearest Neighbors algorithm created a model with balanced data by using SMOTE or the oversampling technique. As a result, the model was able to better classify the datasets because the data balancing had prevented biased results from occurring when the data was imbalanced. But it is not an optimal solution to address the imbalance challenge as the new data are not real but synthetic. Also, it does not improve performance in the original dataset having class imbalance. Ensembling gives better predictive performance by combining the predictions from multiple models. Bagging, Stacking and Boosting are three methods and we used a boosting method since it is more effective than bagging and less complex than stacking. XGBoosting is a type of boosting algorithm that works by iteratively training decision trees to correct the errors of the previous trees to minimize the loss function using gradient descent. It works on improving loss function and correcting misclassification. [5] Thyroid dataset was fed to three separate algorithms namely Random Forest, SVM and XGBoosting. The models were implemented, and result comparison has been done based on parameters like the F1-score, Accuracy, Precision and Recall. It had been observed that XGBoosting had outperformed the other models while comparing using ROC curve. While this method has a good quality of reducing misclassification in minority class data but it is not optimal for real world problem as it depends on loss function which requires the true label of test data. Real world problems won't have the outcome to compute loss function.

IV. FUZZY SVM

Fuzzy Support Vector Machine is a new classification method that incorporates fuzzy logic in SVM (Support Vector Machine) algorithm to help detect data having class imbalance problem. In SVM, each data is treated equally but for an imbalanced dataset like ours this can be a problem as the minority class data don't get enough attention. As a result, the performance is poor. Fuzzy SVM incorporates Fuzzy logic in classification to handle uncertain data and increase certainty. It gives fuzzy membership to each data that decides the influence of that data on the prediction. [3] The SVM soft-margin optimization problem is reformed using fuzzy weights:

$$\begin{aligned} &\text{Min}\left(\frac{1}{2} w \cdot w + C \sum_{i=1}^{l} m_i \varepsilon_i\right) \\ &\text{s.t.}\, y_i (w \cdot \Phi(x_i) + b) \geq 1 - \varepsilon_i \\ &\varepsilon_i \geq 0, \qquad i = 1, \ldots l. \end{aligned} \tag{8}$$

Here, membership function is incorporated into the objective function to adjust data importance. In Fuzzy SVM this optimization is transformed into:

$$\text{Max}W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$\text{s.t.} \sum_{i=1}^{l} y_i \alpha_i = 0, 0 \leq \alpha_i \leq m_i C, \qquad i = 1, \ldots, l.$$

(9)

To enhance the performance of an imbalance dataset, higher membership is assigned to minority class data in the training set. From existing Fuzzy SVM models, we obtain several fuzzy membership methods. [3] In the existing FSVM-CIL method, membership values are assigned for training examples in such a way to satisfy the following two goals:

1) to suppress the effect of between class imbalance.
2) to reflect the within-class importance of different training examples in order to suppress the effect of outliers and noise. These membership functions as follows:

$$m_i^+ = f(x_i^+) r^+$$
$$m_i^- = f(x_i^-) r^-$$

(10)

Here, m = membership value, f(xi) = fuzzy membership function for xi train sample and r = ratio. To assign higher membership to minority class, r = 1 for minority class and r = minority to majority ratio for majority class. f(xi) is assigned using different methods:

Based on the Distance from the Own Class Center: In this method, f(xi) is defined with respect to the distance between xi and its own class center. The examples closer to the class center are treated as more informative and assigned higher f(xi) values, while the examples far away from the center are treated as outliers or noise and assigned lower f(xi) values.

Distance is measured by the Euclidian distance method:

$$d_i^{\text{cen}} = \|x_i - \bar{x}\|^{1/2}$$

(11)

Linear function adding Δ a small positive value to avoid 0 on the normalized distance value:

$$f_{\text{lin}}^{\text{cen}}(x_i) = 1 - \frac{d_i^{\text{cen}}}{\max(d_i^{\text{cen}}) + \Delta}$$

(12)

Exponential function including β which determines steepness of the decay:

$$f_{\text{exp}}^{\text{cen}}(x_i) = \frac{2}{1 + \exp(\beta d_i^{\text{cen}})}$$

(13)

Based on the Distance from the Actual Hyperplane: In this method, we define f(xi) based on the distance from the actual separating hyperplane to xi, which is found by training a conventional SVM model on the imbalanced dataset. The examples closer to the actual separating hyperplane are treated as more informative and assigned higher membership values, while the examples far away from the separating hyperplane are treated as less informative and assigned lower membership values.

Distance of data from hyperplane:

$$d_i^{\text{hyp}} = y_i(w \cdot \Phi(x_i) + b). \tag{14}$$

Linear and exponential functions:

$$f_{\text{lin}}^{\text{hyp}}(x_i) = 1 - \frac{d_i^{\text{hyp}}}{\max(d_i^{\text{hyp}}) + \Delta}$$
$$f_{\text{exp}}^{\text{hyp}}(x_i) = \frac{2}{1 + \exp(\beta d_i^{\text{hyp}})} \tag{15}$$

These existing membership functions used distance from class center which did not focus on data that are highly risky to be misclassified. Hyperplane based distance performed very well for an imbalance dataset but it used an SVM classifier which made the computation complex. Limitations of these methods are:

- **Class Center:** The method calculates distances from each sample to its respective class center and normalizes these distances. However, it treats each class's distance equally without considering the imbalance in class distributions. In heavily imbalanced datasets, this can lead to disproportionate contributions to membership values from the majority class, potentially overshadowing the minority class.
- **Hyperplane:** SVM classifier is mainly used for classification and to improve imbalance learning fuzzy membership is added. The method heavily relies on the performance and output of the SVM classifier. If the SVM model does not perform well (e.g., due to overfitting, underfitting, or inappropriate kernel choice), it can affect the quality of the distance calculations and, consequently, the fuzzy membership values.

V. PROPOSED APPROACH

Our proposed approach to assign fuzzy membership tends to overcome these limitations. This method assigns fuzzy membership based on the distance of training data from the threshold value of the two most significant features. This way of assigning membership can improve data importance as it is derived from features that affects the outcome most. The significant features were selected from correlation coefficient using Pearson, Spearman and Kendall method. Then the threshold values of those features were obtained from the data distribution that shows the class separation range. Precisely, mean value of features can give an optimal threshold which was used in our proposed approach. This approach overcomes the limitation of hyperplane-based membership as it is nor complicated neither dependent on any complex model. Also, it focuses on the data close to threshold value or class separation line where the class center-based membership focused on data close to each class. That means this approach overcomes the limitation of class center-based membership. Distance from the feature threshold is defined as:

$$f_{\text{distance}}[i] = |T - X_{fi}|$$

(16)

It is the distance from T, threshold value of the feature to ith train data of the feature f.

Linear function adding Δ a small positive value to avoid 0 on the normalized distance value:

$$f_{\text{lin}}(x_i) = \frac{f_{\text{distance}}(i)}{\max(f_{\text{distance}})+\Delta}$$

(17)

Exponential function including γ parameter that controls the steepness of the exponential function:

$$f_{\exp}(x_i) = \frac{2}{1+\exp(\gamma \cdot f_{\text{distance}}(i))}$$

(18)

The combined membership of both features:

$$f(x_i) = \alpha \cdot f_1(x_i) + \beta \cdot f_2(x_i)$$

(19)

α and β are weights assigned to the fuzzy membership functions for both features. The final membership assigned to data in terms of their class:

$$m_i^+ = f(x_i^+)r^+$$
$$m_i^- = f(x_i^-)r^-$$

(20)

r is defined by the ratio of minority to majority data. This parameter increases the membership of negative data.

VI. METHODOLOGY

Data Collection: The dataset used for the research is the “Thyroid Disease Patient” dataset which captures various attributes related to thyroid conditions for medical diagnosis. It includes demographic information such as age and sex. Medical history features encompass intake of thyroxine, antithyroid medications, and past surgeries. Patients' current health status regarding sickness, pregnancy, presence of goiter, tumor, or hypopituitary conditions is recorded. Additionally, it notes whether patients suspect hypothyroidism or hyperthyroidism. Laboratory results like TSH, T3, TT4, T4U, and FTI levels are included if measured. Result denotes the presence or absence of hyperthyroidism. Referral sources are indicated as well. This dataset is collected from Kaggle.com, owner named Prakhar Kapoor. Table I includes all the attributes of the thyroid dataset and Table II shows dataset characteristics:

TABLE I. DATASET ATTRIBUTES TABLE

NO	ATTRIBUTE	NO	ATTRIBUTE	NO	ATTRIBUTE
1	age	11	query_hyperthyroid	21	TT4_measured
2	sex	12	lithium	22	TT4
3	on_thyroxine	13	goitre	23	T4U_measured
4	query_on_thyroxine	14	tumor	24	T4U
5	on_antithyroid_meds	15	hypopituitary	25	FTI_measured
6	sick	16	psych	26	FTI
7	pregnant	17	TSH_measured	27	TBG_measured
8	thyroid_surgery	18	TSH	28	TBG
9	I131_treatment	19	T3_measured	29	referral_source
10	query_hypothyroid	20	T3	30	Result

TABLE II. DATASET CHARACTERISTICS TABLE

Dataset Characteristics:	**Attribute Characteristics:**	**Number of Instances:**	**Number of Attributes:**
Multivariate, Imbalanced	Numerical: 7 & Categorical: 23	3770	30

Data Preprocessing: 7 attributes have been dropped due to irrelevancy. ‘TBG’ attribute was an empty column and an attribute without data is unnecessary. 'TSHmeasured', 'T3measured', 'TT4measured', 'T4Umeasured', 'FTImeasured', 'TBGmeasured' attributes contained unnecessary information. Since we can know the value from the test attributes ‘TSH’, ‘T3’, ‘TT4’, ‘T4U’, ‘FTI’, whether tests were taken is not needed. ‘Age’ attribute has a faulty data 455 so we dropped that data also. ‘Sex’ attribute contained 150 missing values. We filled the missing cells using mode as the attribute is categorical. 63 duplicate data have been dropped. Result attribute has two classes: ‘P’ and ‘N’ with 3381 Positive values and 291 Negative values which refers to 92.1% and 7.9% ratio respectively. That means our dataset is highly imbalanced where the Positive class is dominating. Table III shows data characteristics after preprocessing.

TABLE III. DATASET CHARACTERISTICS TABLE AFTER PREPROCESSING

Dataset Characteristics:	**Attribute Characteristics:**	**Number of Instances:**	**Number of Attributes:**
Multivariate, Imbalanced	Numerical: 6 & Categorical: 17	3672	23

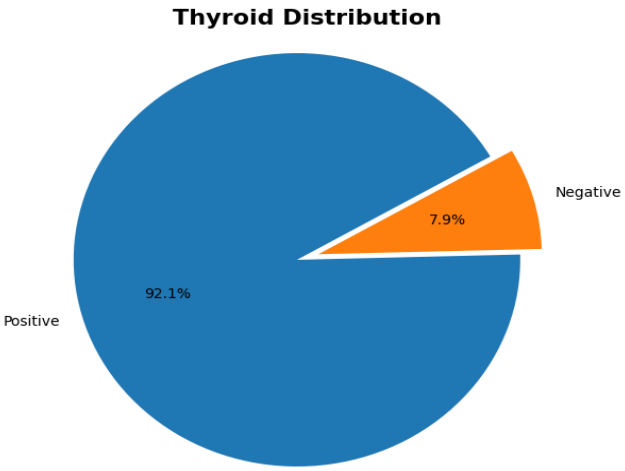

Figure 2: Thyroid dataset class distribution

Feature Selection for Fuzzy Membership: To calculate distance based fuzzy membership, two most significant features of the thyroid dataset were selected. The selection was made by finding correlation between numerical features and result. Pearson, Spearman and Kendall correlation methods were used to find correlation coefficient:

- **Pearson Coefficient:** Pearson correlation coefficient (PCC) is a correlation coefficient that measures linear correlation between two sets of data. Using feature and result mean value and standard deviation, the coefficient r can be defined as:

$$r = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{(n-1)s_x s_y} \tag{21}$$

- **Spearman Rank Coefficient:** It measures the strength and direction of the association between two ranked variables. The coefficient r can be defined as where R means the rank of data:

$$r_s = 1 - \frac{6\sum_{i=1}^{n}(Rx_i - Ry_i)^2}{n(n^2-1)} \tag{22}$$

- **Kendall Coefficient:** It is measured by the similarity of the orderings of the data when ranked. Using C as concordant pairs and D as discordant pairs, the coefficient τ can be defined as:

$$\tau_a = \frac{C - D}{C + D} \tag{23}$$

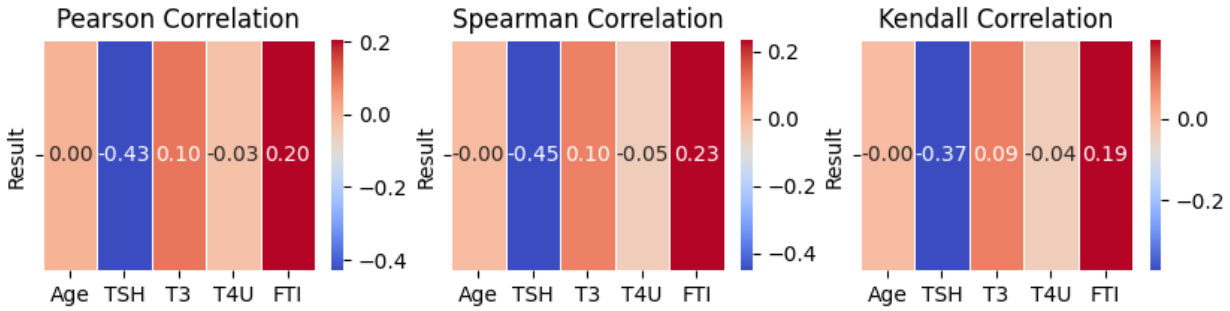

Figure 3: Thyroid dataset class distribution

The most significant attribute is 'TSH' with a correlation coefficient of -0.43(Pearson), -0.45(Spearman). and -0.37(Kendall). The second most significant attribute is 'FTI' with a correlation coefficient of 0.2(Pearson), 0.23(Spearman). and 0.19(Kendall). 'TSH' and 'FTI' were selected as significant features to use in fuzzy membership. In this way the significant features can directly contribute in finding important data that affects the outcome and reduce misclassification.

Threshold Value (T): Threshold value is crucial for calculating feature-based distance since it acts as a separating line between classes. This separating line can be understood from distribution of data in significant features:

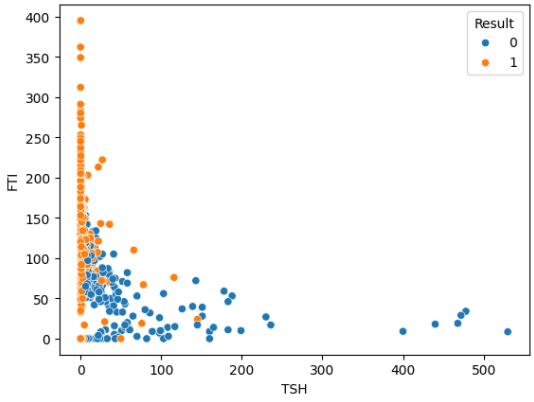

Figure 4: Data distribution for 'TSH' vs 'FTI'

It can be seen that at certain point of both features, data is to be classified in any class. So, threshold value helps determining highly risky data that can be easily misclassified. For more accurate value, mean value of both features 'TSH' and 'FTI' were used as threshold.

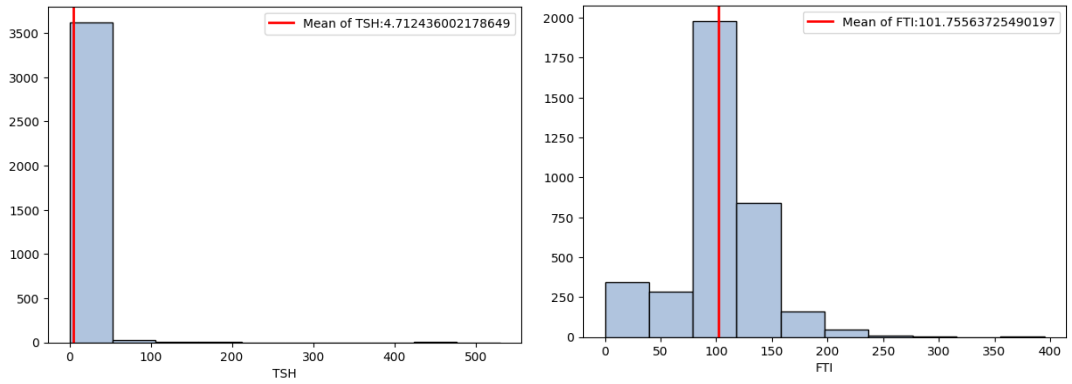

Figure 5: Mean value of 'TSH' and 'FTI' features

Fuzzy SVM using Feature Based Fuzzy Membership: After selecting significant features and their respective threshold values, distance was calculated using (16). Both linear (17) and exponential (18) functions were used to calculate fuzzy membership function for both features. Final function was obtained by combining fuzzy membership function of both feature with their contribution value α and β using (19). α was set to 1 as 'TSH' is the most significant feature and β was the average ratio of 'TSH' and 'FTI' correlation coefficient found in all methods. It was used to adjust the contribution of features according to their significance. Finally, membership was assigned using (20) where r was minority to majority class data ratio for positive class and it was set to 1 for negative class to increase minority class importance and reduce misclassification. Fuzzy membership was then used to adjust weight of data points and implement a new SVM model that includes the power of fuzzy logic. Stratified cross validation was used to maintain same class ratio in all folds and obtain precise result. Fuzzy SVM model was constructed using 'rbf' kernel and C was set to 100 as it showed best result, 'scale' gamma was used in the model.

Evaluation Metrics: To evaluate performance of the Fuzzy SVM model precision, recall and f1 score were used as metrics:

- Precision measures the accuracy of predictions. It is defined as:

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}} \tag{24}$$

- Recall calculates ability of model to predict correctly. It is formulated as:

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}} \tag{25}$$

- F1 score combines precision and recall into a single number using their harmonic mean:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{26}$$

Input:
Dataset {(xi, yi), i = 1, . . . , n}
Output:
The predicted labels of test data
Procedure:
1. Split the dataset into a training set x_train and a testing set x_test using stratified k-fold cross-validation where k=5. For k=1, 2,.., 5 follow the steps
2. Assign fuzzy membership to train data points using (20)
3. Train the Fuzzy SVM model and fit train data into the model with adjusted weights using (8)
4. Obtain predictions of x_test set using the model
5. Evaluate the model predictions by calculating accuracy of each fold and obtain average accuracy
6. Determine confusion matrix of each fold and get the classification report including precision, recall and f1 score for both classes using (24), (25) and (26)
7. Print output results
8. End

ALGORITHM I. FUZZY SVM USING FEATURE-BASED FUZZY MEMBERSHIP

Implementation: Algorithm I described the overall algorithm of Fuzzy SVM model and the code is given below:

```
class FuzzySVM:
    def __init__(self, C, kernel, gamma):
        self.C = C
        self.kernel = kernel
        self.svm = SVC(C=self.C, kernel=self.kernel)

    def fit(self, X, y, fuzzy_membership):
        self.X = X
        self.y = y
        self.fuzzy_membership = fuzzy_membership
        # Weighted fitting
        sample_weights = self.C * self.fuzzy_membership
        self.svm.fit(self.X, self.y, sample_weight=sample_weights)

    def predict(self, X):
        return self.svm.predict(X)

    def score(self, X, y):
        return self.svm.score(X, y)
```

```python
# Define fuzzy membership function
def fuzzy_membership(X,y):
    f1='TSH'
    f2='FTI'
    T1= np.mean(X[f1])
    T2= np.mean(X[f2])
    delta=0.000001
    alpha=1
    beta=0.5
    gamma=1
    f1_distance = np.zeros_like(y, dtype=float)
    f2_distance = np.zeros_like(y, dtype=float)
    for i in range(len(X)):
        # Calculate distance for each row
        f1_distance[i] = np.abs(T1 - X[f1].iloc[i])
        f2_distance[i] = np.abs(T2 - X[f2].iloc[i])

    # #Normalize distance
    f1_distances = f1_distance / (np.max(f1_distance)+delta)
    f2_distances = f2_distance / (np.max(f2_distance)+delta)

    # Using linear function
    f1_membership = (1-f1_distances)
    f2_membership = (1-f2_distances)
    # # Using exponential decay
    f1_membership = 2/(1+np.exp(gamma*f1_distance))
    f2_membership = 2/(1+np.exp(gamma*f2_distance))

    # Assign membership
    membership = (alpha*f1_membership) + (beta*f2_membership)

    # Calculate class ratios
    class_0 = np.sum(y == 0)
    class_1 = np.sum(y == 1)

    # Adjust for class imbalance
    class_0_ratio = 1
    class_1_ratio = class_0/class_1
    membership[y == 0] *= class_0_ratio
    membership[y == 1] *= class_1_ratio

    return membership
# Initialize the Fuzzy SVM model
fuzzy_svm = FuzzySVM(C=100, kernel='rbf', gamma='scale')

# Initialize a StratifiedKFold for 5 splits/folds
skf = StratifiedKFold(n_splits=5, random_state=1, shuffle=True)
```

```
# Placeholder for model accuracy scores and metrics
accuracy_scores = []
conf_matrices = []
classification_reports = []

# Use a for loop and StratifiedKFold's split operation to
# get the train and test row indexes for each split
for i, (train_index, test_index) in enumerate(skf.split(x, y), 1):
    # Use these indexes to split the data into train and test dataframes
    x_train_fold, x_test_fold = x.iloc[train_index], x.iloc[test_index]
    y_train_fold, y_test_fold = y.iloc[train_index], y.iloc[test_index]
    # x_train_fold = pd.DataFrame(x_train_fold)
    fuzzy_membership_fold = fuzzy_membership(x_train_fold,y_train_fold)

    # Train the Fuzzy SVM model
    fuzzy_svm.fit(x_train_fold, y_train_fold, fuzzy_membership_fold)

    # Predict on the test set
    y_pred = fuzzy_svm.predict(x_test_fold)

    # Evaluate the model
    accuracy = accuracy_score(y_test_fold, y_pred)
    conf_matrix = confusion_matrix(y_test_fold, y_pred)
    classification_rep = classification_report(y_test_fold, y_pred)
    conf_matrices.append(conf_matrix)
    classification_reports.append(classification_rep)

# Print average accuracy across all folds
avg_accuracy = sum(accuracy_scores) / len(accuracy_scores)
print(f'\nAverage Accuracy: {avg_accuracy:.2%}')

for i, (conf_matrix, classification_rep) in enumerate
 (zip(conf_matrices, classification_reports), 1):
    print(f'\nFold {i}:')
    plt.figure(figsize=(5, 4))
    sns.heatmap(conf_matrix.T, annot=True, cmap='Blues',
                fmt='d', xticklabels=['Negative', 'Positive'],
                yticklabels=['Negative', 'Positive'])
    plt.xlabel('True Label')
    plt.ylabel('Predicted Label')
    plt.title('Confusion Matrix')
    plt.show()
    print(f'\nFold {i} Classification Report:\n{classification_rep}')
```

Figure 6: Python implementation of Fuzzy SVM

VII. RESULT & DISCUSSION

TABLE IV. CLASSIFICATION RESULTS (%) OF EXISTING MODELS

Classifier	Accuracy	Class	Precision	Recall	F1 Score
SVM	96.19%	0	0.84	0.64	0.77
		1	0.97	0.99	0.99
FSVM(Center-lin)	**96.49%**	0	0.71	0.95	0.81
		1	0.99	0.97	0.98
FSVM(Center-exp)	93.93%	0	0.61	0.76	0.66
		1	0.98	0.96	0.97
FSVM(Hyper-lin)	96.38%	0	0. 70	0.95	0.81
		1	0.99	0.96	0.98
FSVM(Hyper-exp)	96.43%	0	0.71	0.94	0.81
		1	0.99	0.97	0.98

TABLE V. RESULT COMPARISON OF PROPOSED MODEL WITH BEST EXISTING MODEL

Classifier	Accuracy	Class	Precision	Recall	F1 Score
FSVM(Center-lin)	96.49%	0	0.71	0.95	0.81
		1	0.99	0.97	0.98
FSVM(Feature-lin)	96.60%	0	0.72	0.94	0.81
		1	0.99	0.97	0.98
FSVM(Feature-exp)	**97.11%**	0	0.75	0.96	0.84
		1	0.99	0.97	0.98

From Table IV, it is seen that FSVM (Center-lin) had the best accuracy (96.49%) but the same approach worked worst in exponential function (93.93%). On the other side, performance of both functions of hyperplane-based membership were good. Comparison of our proposed feature-based Fuzzy SVM in Table V showed better result than the existing model. FSVM (Feature-exp) performed best (97.11%) but both linear and exponential function outperformed the best existing Fuzzy SVM model. Precision, recall and f1 score was highest in exponential function of proposed feature-based Fuzzy SVM classification model. In the linear function, precision was higher than existing best Fuzzy SVM model but recall was slightly less. However, f1 score was equal to the class center-based Fuzzy SVM model.

VIII. CONCLUSION

In this paper, we proposed an improvised version of Fuzzy SVM model which used feature-based fuzzy membership to address class imbalance problem and reduce the affect of this problem. A highly imbalanced thyroid disease dataset was used to fit into the proposed method. The dataset contained 7.9% negative class data which was the minority class. The aim of this proposed model was to improve performance of minority class and obtain better evaluation results than existing Fuzzy SVM models. Two most significant features 'TSH' and 'FTI' were used for this new fuzzy membership method with their contribution parameters α and β. This membership focused on the data close to feature threshold value obtained from mean and increased their membership. Besides, higher membership was assigned to minority class data to minimize the impact of imbalance problem. This approach demonstrated better results from existing models including traditional SVM and Fuzzy SVM. Feature-based Fuzzy SVM model displayed best performance in exponential function. Overall performance was enhanced in each of the functions. For future work, this feature-based fuzzy membership of Fuzzy SVM model can be used in other imbalance datasets consisting more outliers.

IX. REFERENCE

[1] AK. Niazi, S. Kalra, A. Irfan, A. IslaM, "Thyroidology over the ages." Indian J Endocr Metab 2011;15: S121-6.

[2] H. Kumar, "A Novel Approach of SVM based Classification on Thyroid Disease Stage Detection." IEEE Xplore Part Number: CFP20P17-ART; ISBN: 978-1-7281-5821-1.

[3] R. Batuwita, V. Palade, "FSVM-CIL: Fuzzy Support Vector Machines for Class Imbalance Learning." IEEE TRANSACTIONS ON FUZZY SYSTEMS, VOL. 18, NO. 3, JUNE 2010.

[4] W. Chaipanha, P. Kaewwichian, "Smote vs. Random Undersampling for Imbalanced Data- Car Ownership Demand Model." University of Zilina, C.2022.3, Vol 24, D105-D115.

[5] P. Kumari, B. Kaur, M. Rakhra, "Explainable artificial intelligence and machine learning algorithms for classification of thyroid disease." Discover Applied Sciences (July 2024), 6:360.

[6] K. Yang, Z. Yu, X. Wen, W. Cao, C. L.P. Chen, H.S. Wong and J. You, "Hybrid Classifier Ensemble for Imbalanced Data." IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, 2162-237X © 2019 IEEE.

[7] C. Wang et al. "Adaptive ensemble of classifiers with regularization for imbalanced data Classification." Information Fusion 69 (2021) 81–102.

[8] X. Gu, T. Ni and H.Wang, "New Fuzzy Support Vector Machine for the Class Imbalance Problem in Medical Datasets Classification." The Scientific World Journal Volume 2014, Article ID 536434.

[9] S. T. Lim, K.W. Khaw1, X. Y. Chew and W. C. Yeong, "Prediction of Thyroid Disease using Machine Learning Approaches and Featurewiz Selection." Journal of Telecommunication, Electronic and Computer Engineering, e-ISSN: 2289-8131 Vol. 15 No. 3.