TABLE OF CONTENTS

ABBREVIATIONS

LJMU	Liverpool John Moores University
FRAT	Fall Risk Assessment Tool
SMOTE	Synthetic Minority Oversampling Technique
ANN	Artificial Neural Network
SVM	Support Vector Machines
GANN / GAN	Generative Adversarial Neural Networks
MLP	Multilayer Perceptron
DCGAN	Deep Convolutional Generative Adversarial Networks
LSTM-GAN	Fusion model of long short term memory network and generative adversarial network
ESRGAN	Enhanced super resolution generative adversarial networks
CNN	Convolutional Neural Networks
CTGAN	Conditional Tabular Generative Adversarial Networks
TGAN / tabGAN	Tabular Generative Adversarial Networks
SHAP	Shapley Additive Explanations
FCA	Foot Contact Area
A_fall	Ascent Falls (True/False – 1/0)
D_Fall	Descent Falls (True/False – 1/0)
Combined_fall	Combined Falls (True/False – 1/0) – Either of ascent/descent falls
CDF	Cumulative Distribution Function
ML	Machine Learning
LR	Logistic Regression
LR_CV / LRCV	Logistic Regression with Cross validation
SVM_CV / SVMCV / SVM	Support Vector Machine with Cross Validation
RF_CV / RFCV / RF	Random Forrest with Cross Validation

XGB_CV / XGBCV / XGB	Extreme Gradient Boosting with Cross Validation
GANN2000	Oversampled data based on 2000 samples of synthetic data generated using CTGAN
GANN1000	Oversampled data based on 1000 samples of synthetic data generated using CTGAN
GANN500	Oversampled data based on 500 samples of synthetic data generated using CTGAN
ROC-AUC	Receiver Operating Characteristic Curve – Area Under the Curve
AP	Average Precision

TABLE OF FIGURES

Table of Tables

1 INTRODUCTION

Fall is a common but often overlooked cause of injury mainly among older people, especially those aged 65 years and over, who are more vulnerable and susceptible to falls, especially if they have certain health conditions. These conditions may be balance problems and muscle weakness, vision loss or long term health problems like heart disease, dementia etc. Some mobility affecting conditions such as arthritis, diabetes, incontinence, stroke, syncope, or Parkinson's disease are also major risk factors for falls in older people [1]. Around 1 in 3 adults over 65 and half of people over 80 will have at least one fall a year [2]. Most falls don't cause serious injuries, however there is always a risk of broken bones and thereby causing person to lose confidence and become withdrawn with a feeling of losing once independence [2], or in extreme cases resulting in deaths . Among all these fall related injuries or deaths, stair fall accounts to 60% of it every year [3].

Multiple fall risk screening tools like FRAT [4] exists, which encompass physical and psychological parameters that are associated with stair negotiation performance with related parameters such as vision, fear of falling, balance confidence levels etc., but are not entirely stair specific and commonly used for general falls. Stair negotiation can be executed using techniques specific to capabilities and deficits of individuals; for example, by negotiating stairs in a step by step manner or by relying on handrails, older adults with balance deficits might be able to minimise stair fall risk. Hence in such cases, biomechanical stair specific testing is necessary to profile individuals stepping strategies (grouping certain individual) based on multiple parameters like cadence, foot clearance etc., reflecting on both risk and safety on stairs [5]. For example, profiles can be created based on cases such as the one reported where, older adults displaying increased variability in foot clearance, which is a risky strategy for a trip, also displayed an increased foot clearance, which minimizes trip risk [5].

One such study was conducted within Liverpool John Moores University (LJMU) – School of Sports and Exercise Science, to investigate whether stair fallers could be differentiated from non-fallers by biomechanical risk factors or physical/psychological parameters and to establish the biomechanical stepping profile posing the greatest risk for a stair fall [6]. This study highlighted the potential of the stepping profiling method to predict stair fall risk in older adults by using survival analysis [7] and K-means clustering [8], however the findings deemed physical/psychological parameters (Non-Stairs data) not useful in predicting hazardous event. Additionally, no specific single parameter responsible for the stair fall risk was determined.

Based on this existing study, the initial objective of this project was to use supervised machine learning approach rather than statistical approach to determine stair fall risk and explore causal relationships between parameters responsible for stair falls. However, during the initial phases, the study was reformed due to the nature of available data which was extensive in terms of features but in limited amount which accounted for less than 100 subjects/instances, and exhibited significant imbalance among the target classes. The research being funded by LJMU Sports and Exercise Science Department, also played a major role in shaping the final objective with emphasis given on Non Stairs data (physical/psychological parameters), which in the previous study was not deemed useful [6].

The study presents application of various supervised machine learning algorithms to predict stair falls, on 2 sets of data – (i) non stairs data for all subjects and, (ii) combination of stairs and non-stairs data for subjects who are non-dependant on hand rails. Additionally, the impact of various data oversampling strategies on machine learning models is demonstrated, with emphasis on Generative Adversarial Neural Network - synthetic data generation technique [9]. Furthermore, how feature selection can influence final outcome is also explored. Feature contributions on the model outputs are also determined.

2 LITERATURE REVIEW

2.1 STUDY ON PREDICTION OF BALANCE PERTURBATIONS AND FALLS

The study conducted within LJMU - Prediction of Balance Perturbations and Falls on Stairs in Older People Using a Biomechanical Profiling Approach [6] is the basis upon which the present project is executed. This study focused on using statistical approach of Kaplan Meier and cox-regression survival analysis techniques along with k-means machine learning algorithm for clustering. The experiment involved 87 older adults who negotiated seven step biomechanical staircase and performed range of physical/psychological tasks. Based on the biomechanical parameters, k-means clustering was used to profile stair negotiation behaviour. Hazardous events which comprised of falls and events of balance perturbation were then monitored during 12-months follow up period. Impact of physical/psychological (Non-stairs data) parameters or biomechanical (stairs related data) outcome measures were examined to predict future hazardous events using cox-regression analysis, with stepping strategies posing a risk for these events identified using Kaplan-Meier survival curves.

As discussed earlier, the results did not deem Physical/psychological parameters useful as it did not predict hazardous events and the commonly used Fall Risk Assessment Tool (FRAT) classified only 1/17 stair fallers at risk for a fall. Additionally, Single biomechanical risk factors could not predict hazardous events on stairs either. Whereas, two particular clusters identified by the stepping profiling method in stair ascent were linked with hazardous events [6].

Figure 1. 7-Step Biomechanical Stair Case at Liverpool John Moores University

The data used for this research is the same one which was being used within the above study, with some additions to of few more subjects (instances/rows). This data being extremely imbalanced (minority class - falls == True) and significantly less in size (number of rows), called for exploration of oversampling and synthetic data generation techniques.

2.2 DATA IMBALANCE AND OVERSAMPLING STRATEGIES

Imbalanced datasets display severe skewness in class distributions such as 20:80 instances in minority and majority classes. This can sometimes influence machine learning algorithms to ignore minority class entirely and raises major issue in the desired outcome which demands need to balance the classes.

Randomly resampling (oversampling / undersampling) the training dataset is one of the approaches that has been known to address the problem of class imbalance [10]. Undersampling is ideal if the data size is large enough as it involves randomly deleting majority class examples. Undersampling effect can be observed in a study where five new imbalanced big datasets, with target positive classes of 10%, 1%, 0.1%, 0.01%, and 0.001%, were created from each existing original full dataset to study information loss introduced in traditional random undersampling and modelled using random forest algorithm; it was noted that random undersampling the negative class to 50:50, showed similarities regarding average performance compared to that of using the entire big dataset [11]. However, this has its drawback if size of data is significantly less as removing instances to balance classes may result in information loss, hence oversampling becomes ideal is such cases.

Random oversampling involves randomly duplicating minority class examples by picking samples at random with replacement. This technique has been found to be effective in ML algorithms that learn iteratively using stochastic gradient descent like ANN, and models like decision trees and SVM which seeks good data splits. One such case is where, random oversampling was used along with random forest algorithm to predict stroke resulting in 99.5% accuracy [12].

However, seeking a balanced distribution for severely imbalance data set using random oversampling can increase the likelihood of overfitting minority class leading to increased generalization error, especially for higher over-sampling rates. Moreover, it may decrease the classifier performance and increase the computational effort [10].

Another oversampling approach known is the synthetic minority oversampling technique (SMOTE), where the minority class in oversampled by generating synthetic examples relatively close in feature space with exiting minority class examples, rather than just duplicating / oversampling with replacement, thereby reducing the risk of overfitting [13]. The technique is known to work well with boosted trees and random forest algorithms. One such study is where, adopting SMOTE with random forest to effectively predict depression among women in Malaysia resulted in improved classification accuracy and sensitivity when compared with using unbalanced data [14]. Another study to detect fraudulent insurance claims reports exceptional results after balancing classes using SMOTE and classifying using random forest [15].

However, the downside here is a possibility of ambiguity in data if there is a strong overlap for classes (sample overlapping), as synthetic data here is created without considering the majority class. Other limitations can be noise interference and blindness of neighbour selection [16].

Generative Adversarial Neural Networks (GANN/GAN) based approach for synthetic data generation is known to overcome few of the shortcomings seen with SMOTE, as it learns from overall class distribution unlike SMOTE which is based on local information. It involves discovering and learning patterns or regularities in input data allowing the model to generate synthetic samples that plausibly could have been drawn from the original data set.

The generative model is executed as a combination of two neural networks – i) Generator, which generates plausible data resembling some known distribution; and ii) Discriminator, which learns to distinguish generators fake data from real data while penalizing the generator for producing implausible results during each training iteration [17]. The generator network updates its weights using backpropogation based on the signals from discriminator and this cycle is repeated until generator produces synthetic data that is similar enough to the real data, and the discriminator can no longer distinguish between the two.

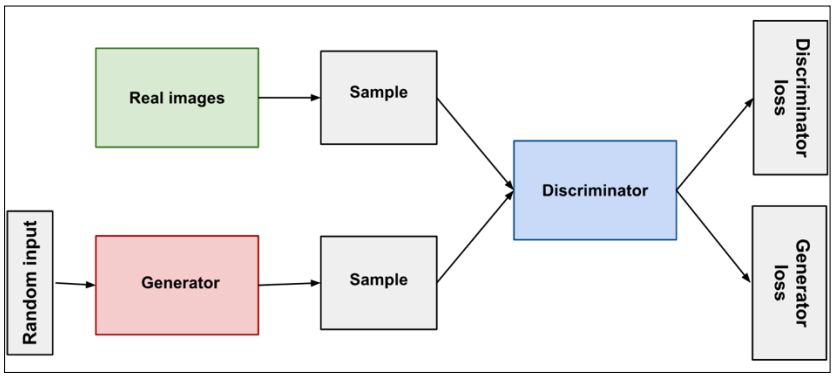


Figure 2. GAN Structure for Synthetic Image Generation [17]

The first GAN based architecture demonstrated use of MLP and CNN, however presently multiple alternative architectures exits to incorporate various types of data [18]. While a research for generating new human faces shows improvement in image quality by employing DCGAN and ESRGAN architectures [19]; another study for time series anomaly detection showed superior performance in processing time series data compared to conventional algorithms, when LSTM-GAN fusion model was used [27].

There have been progressive studies to alleviate class imbalance in tabular data using GAN. A GAN based survey suggests how GANs have showed improved performances over the years in imbalanced tabular datasets compared to traditional baseline techniques like smote [20]. Another research shows how tabular GAN oversampling outperforms traditional techniques when applied on time to event data for survival prediction [23]. There are studies which even compare several architectures of tabular GANs like a research on EEG (electrical activity of brain) data has shown synthetic data generated by conditional tabular GAN (CTGAN) to be more similar to the one generated by traditional tabular GAN (TGAN), with CTGAN getting higher score in basic statistics, correlation column correlations, mean correlation between fake and real columns and similarity scores [22].

GAN based techniques for tabular data generation have their own downsides with it mainly being computationally intensive, and requiring large amount of tabular data to produce similar data with statistical properties in line with real data. Also, the result significantly depends on the quality of the training data sometimes even resulting in overfitting.

Several studies have tried to combine other sampling techniques or several statistical methods with GAN to improve performances such as shown in a research [21] which suggests how even though GAN has produced 10/11 times better results than smote, combining the two (SMOTified-GAN) improved the data quality thereby providing better results.

Every oversampling based studies are specific to certain datasets and using one technique may not always results in same performance with a dissimilar dataset. This project is limited to use of traditional random oversampling, SMOTE and tabular GAN based oversampling technique. Additionally, a study suggesting significant impact of feature selection on model outputs when synthetic data was involved [25], inspired use of univariate feature selection technique and studying its impact, for this project.

3 DATA DESCRIPTION

The available data was provided by LJMU School of Sports and Exercise Sciences. The data was made available as excel workbooks/files (total = 32) with each file having one or more worksheets (total = 62), each with several features. The data contains information on 94 subjects who were introduced to biomechanical staircase. The fall related data was measured over a 12 month follow up period. Each of the worksheets have 'subject_id' column which uniquely represents the subjects involved in the experiment and links the overall data. Note that it follows the ethical guidelines and no personal information is present.

This dataset was available as 3 major categories as follows:

a. Non Stairs Data: This data comprises of non-staircase related measures.
   - It includes 3 files –
     i. 1 workbook for questionnaires and various fall assessment and balance tests with 8 worksheets.
     ii. 2 workbooks for Postural Balance Data with 1 worksheet in each.
   - Questionnaires and Balance Tests – This single file contains the following worksheets (collections of related data) and each with multiple features:
     i. General Info : This includes subjects physical related features like height, weight, gender, age etc. Also, since this data was gathered before the 12 month follow up period, it also includes data which informs if the subject had a general fall or not in the previous 12 months, and if the subject was dependant on handrail while using stairs.
     ii. Vision Test : This includes visual acuity and contrast sensitivity measures, and these are obtained using the Freiburg Vision Test [6] [28].
     iii. Handheld Dynamometer : This data consists of measures of isometric muscle forces of hip flexion, hip abduction, knee flexion, and knee extension which were obtained using a handheld myometer (M500 MyoMeter, Biometrics Ltd, UK) [6] with the maximal torque calculated and normalized for body mass [29].
     iv. Range of Motion Test : This data consists of measures assessed actively and passively for the hip flexion, knee flexion, and ankle plantar and dorsal flexion in the supine position using a goniometer [6] [30].
     v. Functional Test : This data includes measures from Fullerton Functional Fitness Test included as single independent outcome measures: (i) the amount of chair stands performed in 30 seconds, to measure functional lower body strength; (ii) the amount of arm curls per- formed in 30 seconds, to measure functional upper body strength; (iii) the amount of steps performed within a $1minutes step test, to measure physical stamina; (iv) chair sit and reach, to measure lower body flexibility; (v) back scratch test, to measure upper body flexibility; and (vi) timed up and go, to measure speed, agility, and balance [6] [31]. Also, Functional reach test, which provides information related to anticipatory

balance control without a change in the base of support while reaching forward is included [6]. Additionally, this file also consists measures of handgrip strength and lower body strength.

vi. Berge Balance : This data includes balance measures using Berg Balance Scale (BBS), which addresses anticipatory balance control with and without a change in base of support during 14 different functional tasks, in addition to single leg stance test, included in the BBS, as a separate measure and performed for a prolonged period of 30 seconds, while the number of times the participant having had to regain their balance [6] [32].

vii. Star Excursion Balance Test : This data consists of measures of dynamic postural control in the lower limb, where the person performing the test is to maintain single leg stance on one leg while reaching as far as possible with the contralateral leg in 8 different directions [33].

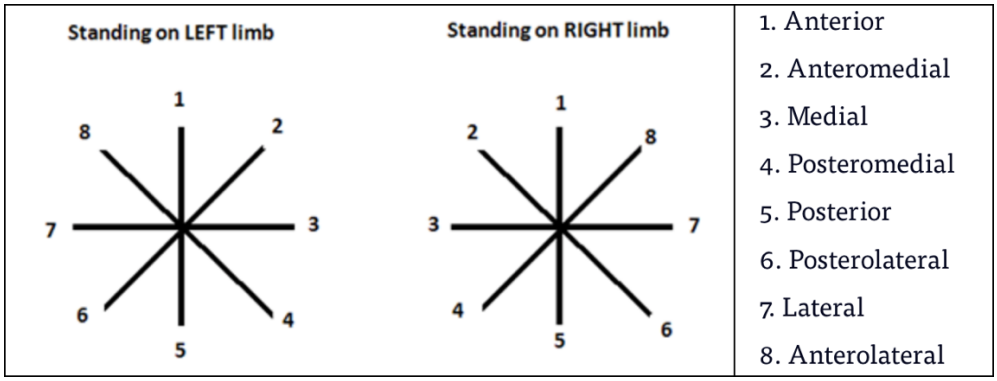


Figure 3. Star Excursion Balance Test Parameters [33]

viii. Questionnaires : The data consists of questionnaires/features involving the Activities- specific Balance Confidence Scale to assess self-perceived balance confidence while performing daily activities; the Falls Efficacy Scale-International to assess concern about falling while completing activities of daily living; the Montreal Cognitive Assessment (MoCA) to assess the cognitive functioning; the Nottingham extended Activities of Daily Living (NADL) index, to assess the functional status; and the FRAT to determine fall risk [6]. This file also included subjects number of medications and falls recorded over previous 1 year before test.

- Postural Balance Measures – This data comprises statistical measures of postural balance data like standard deviation, mean variance etc., obtained on a biomechanical force plate in 2 different settings for subjects – with open eyes and closed eyes.

Features / data from all the above mentioned files were clubbed as non-stairs dataset and used as the independent features for this project.

b. Stairs Data: This data included measures obtained using data analysis performed on participants data (Full Body Kinematics) measured over biomechanical staircase and present in kinematic measure data files.

Additionally, this had been used to obtain biomechanical outcome measures through some data analysis and statistical techniques [6] used predominantly for clustering purpose, and these measures were available for ascent and descent stairs in 2 different staircase settings – easy and normal stairs [6] (total 4 work sheets). These measures include (1) foot clearance, (2)proportion of foot length in contact with stairs (FCA - ascent, Overhang – descent), (3) required coefficient of friction, (4) cadence, (5) maximal CoM (centre of mass) angular acceleration (for decent stairs) and (6) trial to trial variability of all these parameters [6].

Each of these parameters are present in each of these 4 different files indicating -

I. Easy Ascent (up) Staircase
II. Easy Descent Staircase
III. Normal Ascent (up) Staircase
IV. Normal Descent Staircase

This data consists of only subjects who were non dependant on hand rails, and only this subset of data (4 files), clubbed as stairs data set was used as another set of independent features for this study. Remaining kinematic data files were excluded.

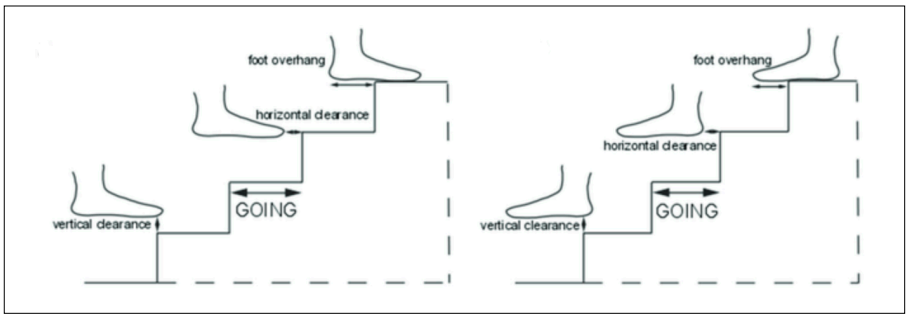


Figure 4. Staircase related measurements (ResearchGate – Stair Gaits in Older Adults)

c. Fall Data : This data present in 1 single file, consists of the falls recorded by subject over the 12 month follow up period, for both ascent as well as descent stair movements. These are recorded such as if the subject had a fall or not (1/0). These 2 features : (1) Ascent Fall – 'A_fall' and (2) Descent Fall – 'D_Fall', have been used as target variables for this study. Additionally, a third target feature (3) Combined Fall – 'combined_fall' has also been used and computed in a way such that a subject who have had either of ascent or descent falls is recorded as true/1 else no fall – false/0.

4 METHODOLOGY

Traditional data analysis approach was applied with a computing pipeline programmed to execute the necessary methods. The below figure depicts a high level overview of the experiment and details discussed in sections ahead.

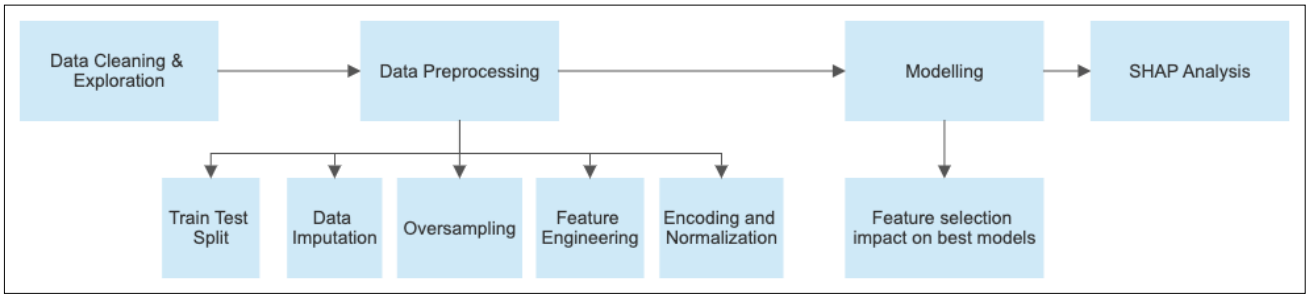


Figure 5. Experiment Overview

4.1 DATA EXPLORATION AND FINDINGS

Having obtained the relevant data which contains both numerical and categorical features, the first step was exploring the data and cleaning and preparing it for the pre-processing stage. Firstly, all the excel files were loaded and for each file, each worksheet was traversed and loaded as a dictionary of dataframes accordingly. All worksheets were loaded as dataframes and the relevant ones were used. Each of these were first individually explored and further merged over 'subject_id' (Table 1). After final merging 2 datasets were produced (Table 2) -

1) Dataset 1 : Non stairs data + Fall data
2) Dataset 2 : Non stairs data + Stairs Data (non- handrail dependant) + Fall data.

Note: The name Dataset 1 has been interchanged with *'Non Stairs Data'* and Dataset 2 interchanged with '*Non Stairs + Stairs Data'* or *'Stairs + Non Stairs Data'* while discussing ahead.

Each of these datasets were further divided based on the 3 fall categories : Ascent, Descent and Combined, when providing input to the model. The entire experiment was first conducted using non stairs dataset and further using non stairs + stairs dataset.

Before merging the data was explored for noise, missing values, outliers and statistical parameters like median, mode, standard deviation etc. to check the data distributions and how spread the data was. The data types of features were also changed accordingly based on categorical and numerical type. The data had both ordinal and nominal categorical variables. The data types of ordinal variables were kept as it is i.e. Integer as categorical encoding these variables ahead could have made it non-meaningful.

Table 1. Relevant datasets used for the study

Available Dataset		File Names (.xlsx)	Total number of files/workbooks		Total number of dataframes/worksheets	
Non Stairs Data	Questionnaires and Balance Tests	DATA_General info_Non-stair tests_Questionnaires	1		8	**10**
	Postural Balance Measures	Postural balance on force plate_Closed Eyes	1	2	1	
		Postural balance on force plate_Open Eyes	1		1	
Stairs Data	Non handrail dependant – stairs – clusterting dataset	DATA_kmeans_Descent_Easy Staircase	1	4	1	**4**
		Data_kmeans_Descent_Normal Staircase	1		1	
		Data_up_clustering_EasyStaircase	1		1	
		Data_up_clustering_Normal Staircase	1		1	
Fall Data	Target Features	Survival_falls_28.03.2019	1		**1**	

Table 2. Merged datasets and available number of subjects

	Merged Datasets	Total dataframes merged	Stair Category	Total subjects present after merge
Dataset 1	Non Stairs Data + Fall Data	11	Ascent	94
			Descent	94
			Combined	94
Dataset 2	Non Stairs Data + Stairs Data + Fall Data	15	Ascent	72
			Descent	70
			Combined	72

Having known the total number subjects, initial step was to check how many subjects have encountered falls throughout the 12 months follow-up period for all 3 fall categories First, the non-stairs dataset was explored and significant class imbalance was discovered with only few subjects encountering fall especially very few descent falls. Additionally, number of subjects who have had general falls in the previous 12 months, prior to the 12 month follow up, were relatively more balanced (Figure 6).

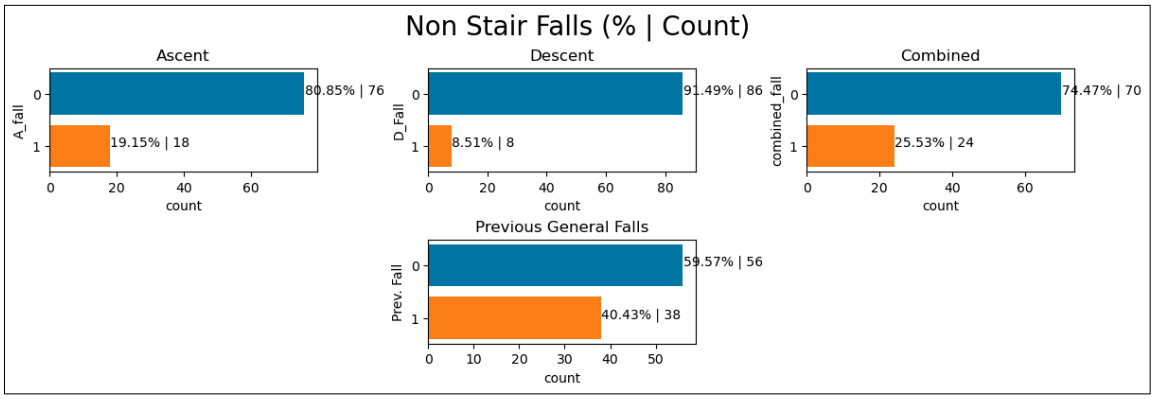


Figure 6. Number of Subjects Reporting Falls (Non Stairs Data)

Similar check was conducted with the non-stairs + stairs dataset and similar imbalance with much lower instances being observed (Figure 7).

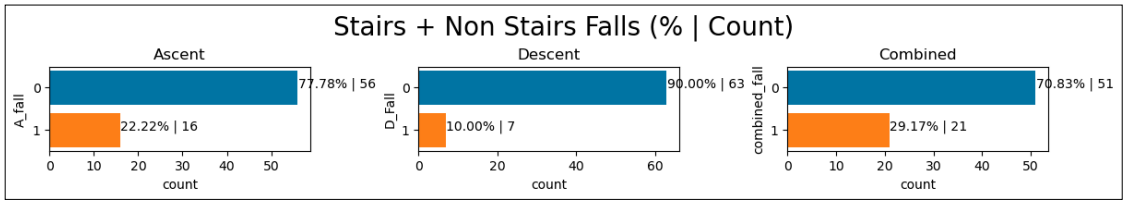


Figure 7. Number of Subjects Reporting Falls (Stairs + Non Stairs Data)

Also, majority of subjects involved in the experiment were observed to be females (Figure 8). Additional exploration was also conducted with non-stairs data with the different fall categories combinations (Figure 9).

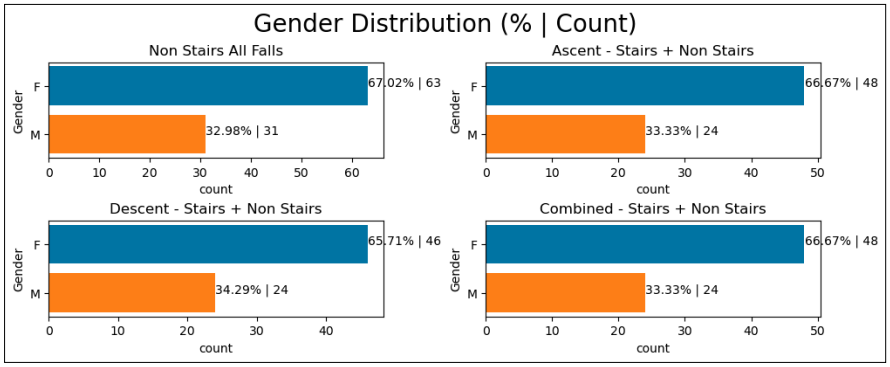


Figure 8. Gender distribution of subjects involved

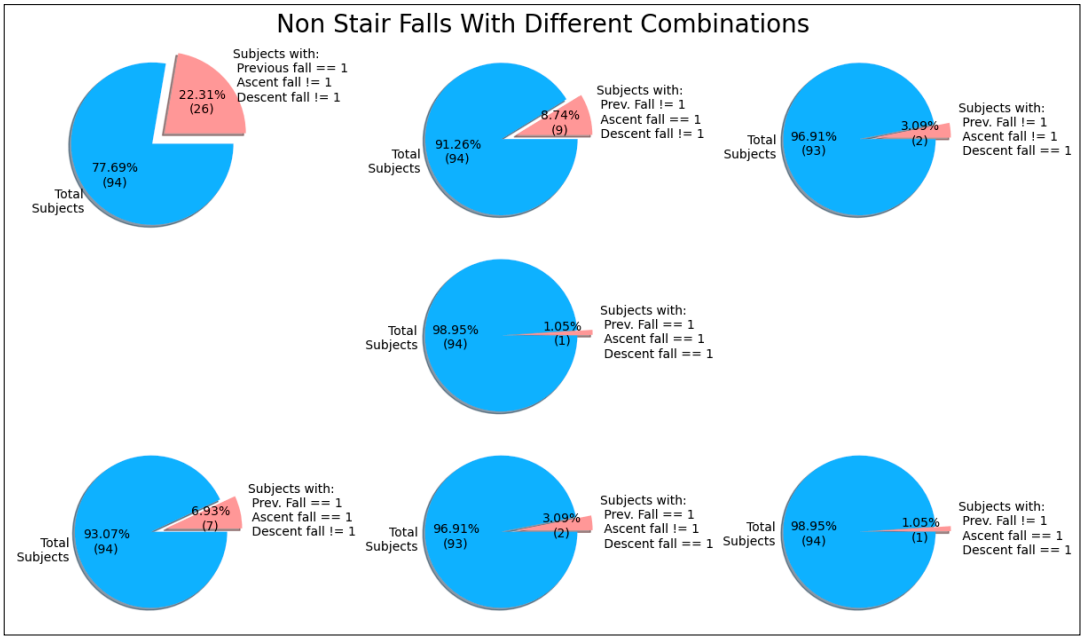


Figure 9. Fall distributions for various combinations of falls

Normality was checked along with linear relationships among parameters of different groups/collection of data using pair plots. Correlation was also found among these parameters and target features, however this was not an ideal approach to check how independent features could affect the dependant, as the target features were categorical. This can be viewed in the 3 figures ahead.

	Hip_ab	Hip_flex	Knee_ext	Knee_flex
count	94.000000	93.000000	94.000000	94.000000
mean	0.876112	0.463621	0.943377	0.590348
std	0.267215	0.199945	0.300581	0.182848
min	0.300604	0.129104	0.302135	0.213854
25%	0.658353	0.312403	0.726088	0.473848
50%	0.852163	0.429536	0.960722	0.571724
75%	1.069575	0.602001	1.140703	0.716015
max	1.474253	1.092552	1.601249	1.097173

Figure 10. Statistical parameters for features from Handheld Dynamometer data

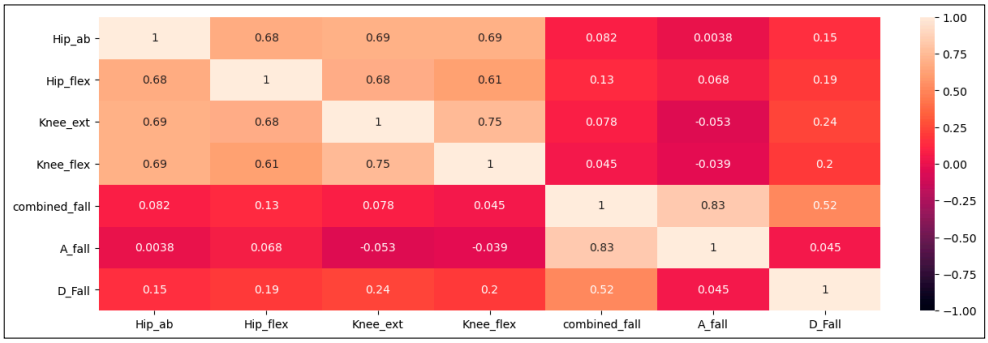


Figure 11. Correlation Heatmap for Handheld Dynamometer features

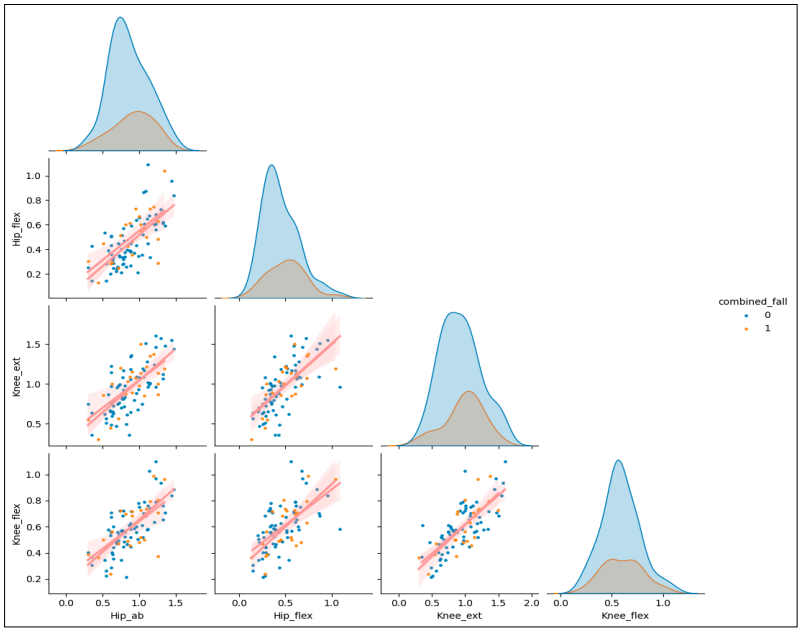


Figure 12. Regression Fit for Handheld Dynamometer Features

As seen in the above figures for Non Stairs dataset – Handheld Dynamometer data collection, a similar process was followed for other groups of data within non-stairs data as well as for features

among stairs + non-stairs data. Features from many individual data collections showed strong linear relationships with each other as can be seen with Handheld Dynamometer field.

Key observations after this exploration was –

a. Severe Imbalance among target classes for both Non stairs and Stairs + Non stairs data observed.
b. Significantly less data available.
c. Some duplicate/redundant columns were eliminated.
d. Missing values were discovered in different features from different groups of data. The instances were not eliminated but rather imputed after train-test split which is discussed in sections ahead.

```
Total rows with missing values: 30
==================================
Shape of the variable - (94, 92)
==================================
nan occurences in total - 212
==================================
Colum-wise occurences:
SV_1                          7
SV_2                          7
SV_3                          7
SV_4                          8
Stereoscopic                  8
Hip_flex                      1
Passive_Knee_flex_degrees     2
Active_Knee_flex_degrees      1
Back stretch (cm)             1
Anterior_01                  20
Anteromedial_01              20
Medial_1                     20
Posteromedial_01             20
Posterior_01                 20
Posterolateral_01            21
Lateral_01                   21
Anterolateral_01             21
MoCa (max. 30)                7
dtype: int64
==================================
```

Figure 13. Missing value occurrences in Non-Stairs Data Set

e. Majority of the features (over 70 %) were normally distributed with few features showing skewness. This observation was necessary to allow us to use an appropriate scaling/normalisation technique in further processing.
f. Collections of data like handheld dynamometer, Star Excursion, ROM, functional test, Postural Balance fields, where features within each of these fields showed strong linear relationships (positive or negative) among each other within the field (like in Figure 12, 14); were significant / had major impact in determining the results i.e. model outputs, as discussed in sections ahead.
g. Linear separability among classes was not observed.

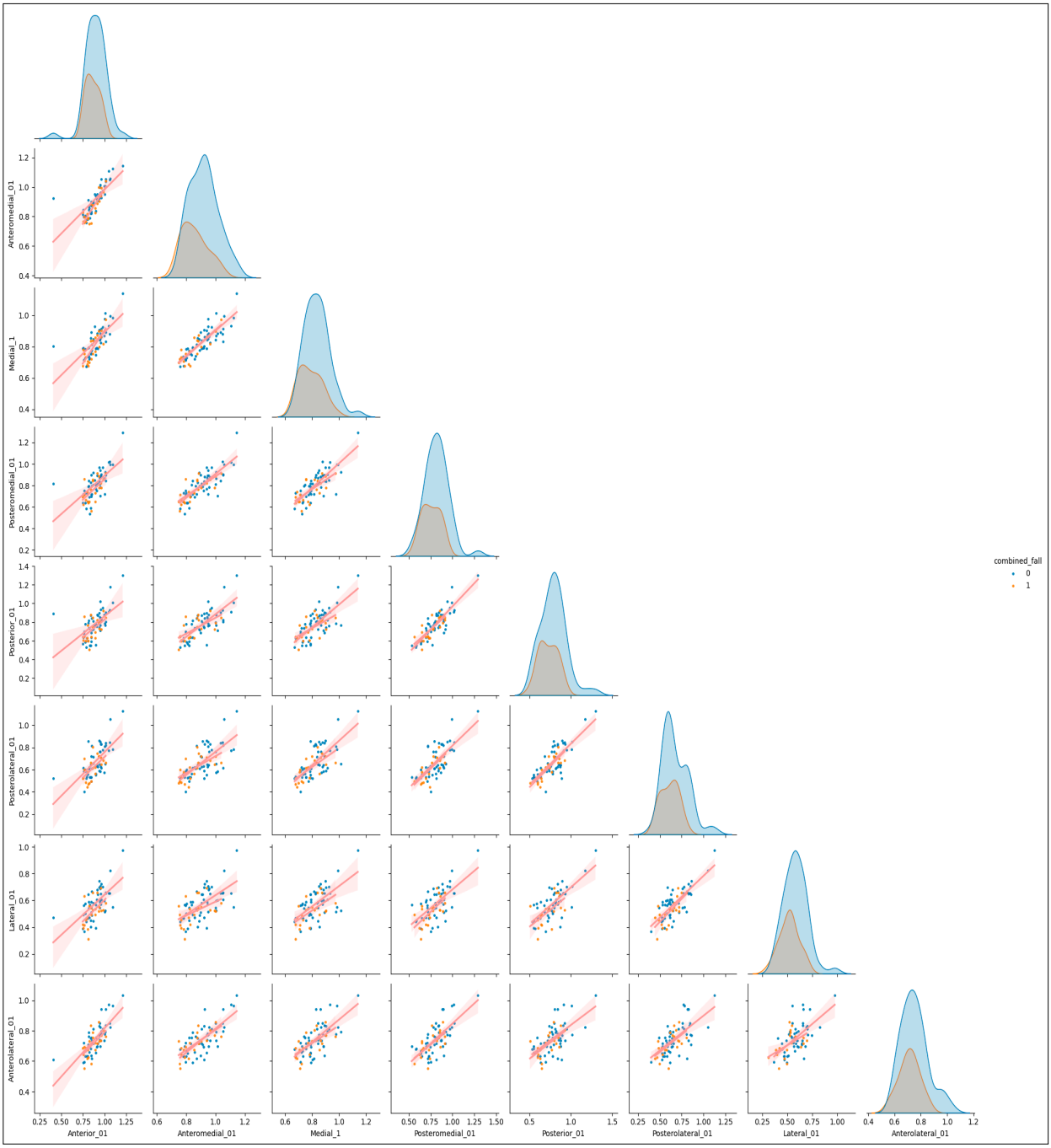


Figure 14. Regression Fit for Star Excursion Test Features

4.2 DATA PREPROCESSING

Since the experiments was conducted for two different datasets, the initial input for the pre-processing stage was as follows:

a. Non Stairs data – This merged data was further split for ascent, descent and combined fall categories (respective targets) and for each these categories the number of instances/subject and total features were same.
b. Non Stairs + Stairs data – Unlike previous dataset, this dataset was supplied to pre-processing stage as 3 subsets of ascent, descent and combined merged along with the respective target variable as each subset of data had different number of subjects and features.
    - Ascent Data
    - Descent Data
    - Combined Data

4.2.1 TRAIN TEST SPLIT

The datasets were after dividing for all the fall categories were then first split so independent (X) and dependant (y) variables were determined (Figure 15).

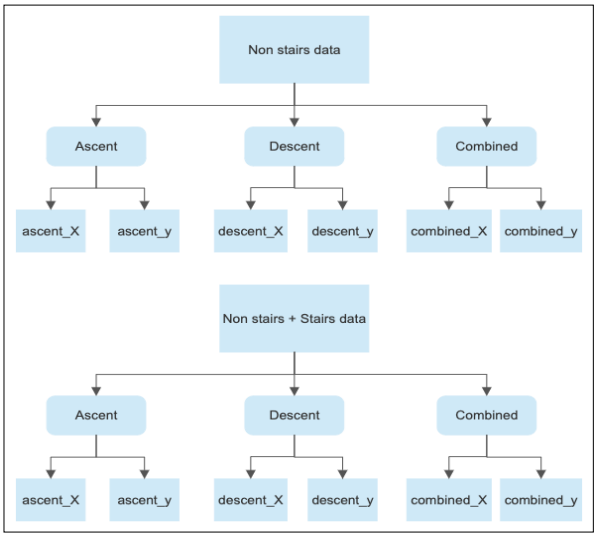


Figure 15. Data Division

Random state was used as it controls the random number generator which is used to shuffle the data before using it. This in turn allows us to get the same data instances each time we call the method. Additionally, stratify was used to maintain same proportion of classes in both the train and test set.

For non-stairs dataset, the experiment was first carried out with 20 % test data split. However to further analyse and to check possible improvements in results, a 30 % test data split was also carried out. For Stairs + Non stairs dataset however, only 30% test data split was performed. For processes

ahead, to avoid data leakage, the methods used train data for fitting to the model/method and the test data was then transformed [34].

4.2.2 DATA IMPUTATION

Imputation was carried out to replace numerical missing values with median rather than mean as mean is susceptible to outliers. Categorical missing values were replaced using mode.

4.2.3 GANN BASED SYTHETIC DATA GENERATION

Before further pre-processing, the entire train (X+y) data was used to generate GANN based synthetic data using CTGAN library [24]. CTGAN provides a GAN based single table data synthesizer which uses a GAN architecture that captures the heterogeneity of data by considering synthetic data generation as a complete flow, from data preparation to the GAN network itself, and handles characteristics of both numeric and categorical features. It introduces mode-specific normalization to handle non-gaussian and multimodal numeric distributions (probability distributions have more than one mode), and conditional generator to handle categorical data along with imbalance [41]. This permits us to skip external normalization and encoding process. Although CTGAN allowed us to add specific feature based conditions while generating data, this project did not use any conditions and the train data was used as a whole (single table with ascent & descent target columns only; combined fall column created on synthetic data after combining the two) was used for the purpose.

This method was used only for non-stairs dataset and number of synthetic samples generated as follows:

Table 3. Synthetic Data Samples

Data	Number of Synthetic Samples
Non Stairs Data – 20% test split	2000
Non Stairs Data – 30% test split	2000
	1000
	500

Once data was generated, the quality of the data was evaluated using synthetic data vault library methods like ‘evaluate_quality’ which provides a *quality score* comparing the original and synthetic data in terms of column shapes and correlations, and, ‘run_diagnostic’ to check if the synthetic data is pure copy of the real data; if it covers the full range of values and if it data adheres to the original ranges [35]. A quality score of about 80% is considered to be acceptable for synthetic data.

To determine the quality score, various metrics [36] are used few of which mentioned as follows –

a. KSComplement : Checks similarity between column shape i.e. marginal distributions of 1D histogram, of real and synthetic column, using Kolmogorov-Smirnov statistic to find maximum difference between 2 CDFs of numerical distributions. Higher score meaning better quality.
b. TVComplement : Check column shape similarity for categorical features. Finds total variation distance using frequency of each category value and expressing it as probability [37].
c. CorrelationSimilarity : This metric measures the correlation between a pair of numerical columns and computes the similarity between the real and synthetic data i.e. it compares the trends of 2D distributions. It supports both the Pearson and Spearman's rank coefficients to measure correlation [38].
d. ContingencySimilarity: This measures the similarity of pair of categorical columns between the 2 data (comparing 2D distribution trend) by using normalised contingency table [39]

Additional quality check was also performed using Table Evaluator which provide visual representations of the real and fake data [40].

While using the synthesizer, the neural network based parameters [24] were adjusted accordingly to get the best quality data. Few parameters discussed ahead (Table 4).

Table 4. CTGAN Synthesizer Parameters

Parameters	Description
epochs	determines number of times required to train the GAN, with possibility of improving the model with each new epoch.
batch_size	Number of samples to process in each epoch. Since original data was less, high batch size was now useful.
discriminator_dim	Determines size of output sample for the discriminator layer. A Linear Layer will be created for each one of the values provided.
embedding_dim	Determines size of random sample passed to the generator.
generator_dim	Determines size of the output samples for each one of the Residuals. A Residual Layer will be created for each one of the values provided.
generator_lr	Learning rate for the generator. Higher learning rate could lead to overfitting.

GANN is stochastic in nature and each time the model is trained, different random samples are generated. Also, generating different sizes of synthetic samples also generates different data as randomness is involve. Hence to reset any randomization in sampling, while generating the samples for 30% test split data, 'reset_sampling' parameter was used.

4.2.4 GANN OVERSAMPLING

Once synthetic data was produced, the synthetic sampling was performed using the original train data. For each of the fall category, respective instances from original train data was used along with random instance from synthetic data. So GANN sampled input data was combination of synthetic data and original data. This can be seen with below example for ascent fall data with 20% split.

Suppose for ascent falls :

- Original train data has 75 subjects/instances,
  - 14 – ascent falls
  - 61 – No ascent falls
- Synthetic data has 2000 instances
  - 307 – ascent falls
  - 1693 No ascent falls

Then, final GANN oversampled ascent input data for the model will comprise of :

- 14 instances from original data where fall is true +
- 307 instances from synthetic data where fall is true +
- 61 instances from original data where fall is false +
- 260 random instances from synthetic data where fall is false.

So final train data for the model will have 321 falls and 321 non falls (equal distribution of classes), so total of 642 instances. Similar, method used for the other two fall categories. The 30% test split data includes similar oversampled data w.r.t 2000, 1000 and 500 synthetic data instances.

4.2.5 ONE HOT ENCODING

Before providing input data to the model, the categorical features were needed to be encoded to numerical values. Nominal categorical variables were converted to indicator variables where, each variable of each feature converted to 0/1 variable.

4.2.6 FEATURE IMPORTANCE

Feature selection involves eliminating redundant and irrelevant data, thereby getting rid of noise in data and this in turn improves the model performance in terms of both learning speed as well as quality of results. Before using the feature selection, it was necessary to know the importance scores of each feature that can reveal which features are relevant to the target outcome. ANOVA F score (f_classif) and Chi-Square scoring functions were considered, however ANOVA was used as several studies showed it to perform better when we have combination of categorical and numerical

features. Also, chi-square is known to produce falsified results when the data values are extremely small which was the case for our datasets. Once the important features were determined using the original train data, feature selection was performed ahead using this importance score. The feature scores for 20% test split non stairs data can be seen below.

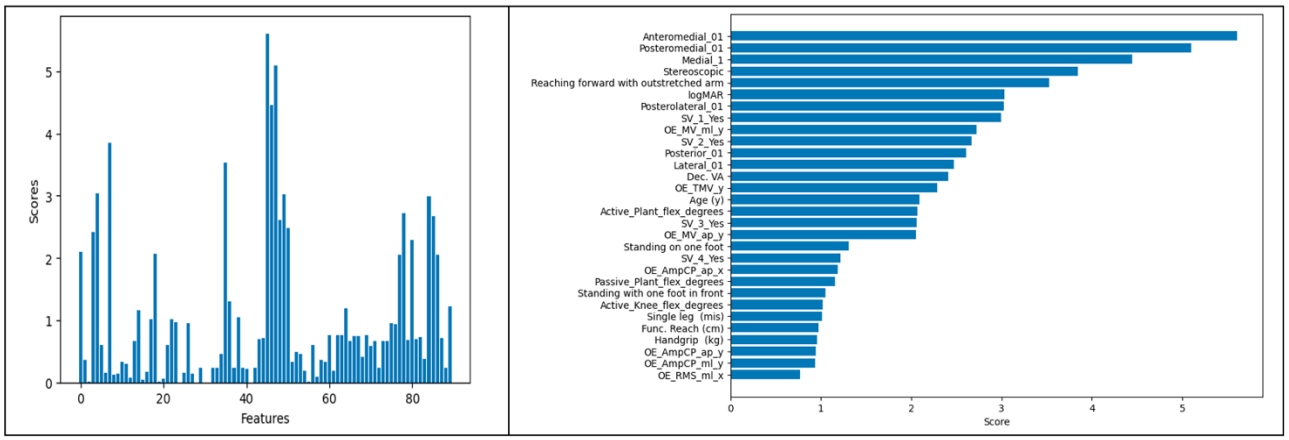


Figure 16. Feature Importance for Ascent Non Stairs data (20% test split)(all - top 30 features)

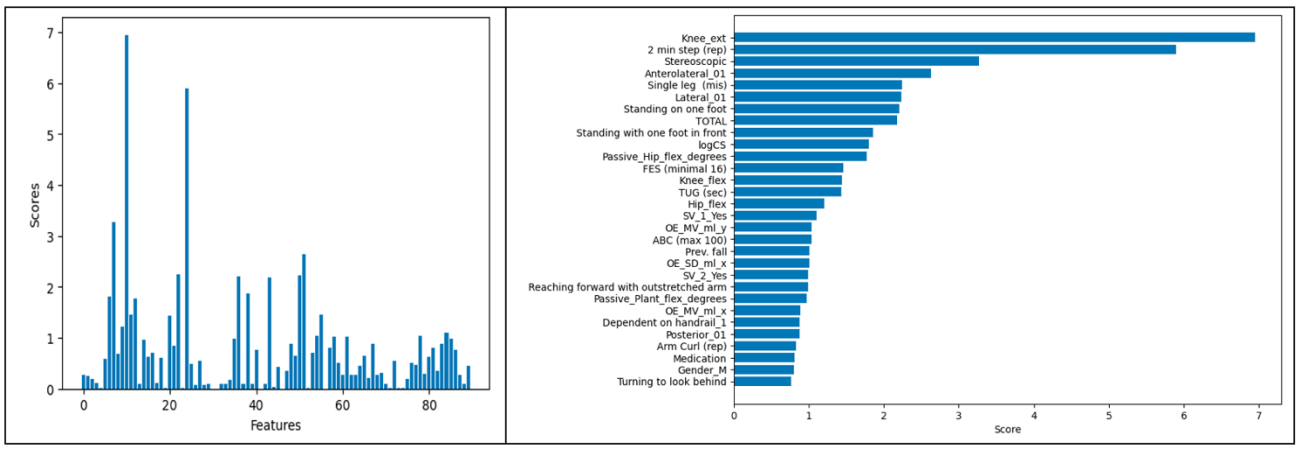


Figure 17. Feature Importance for Descent Non Stairs data (20% test split)(all - top 30 features)

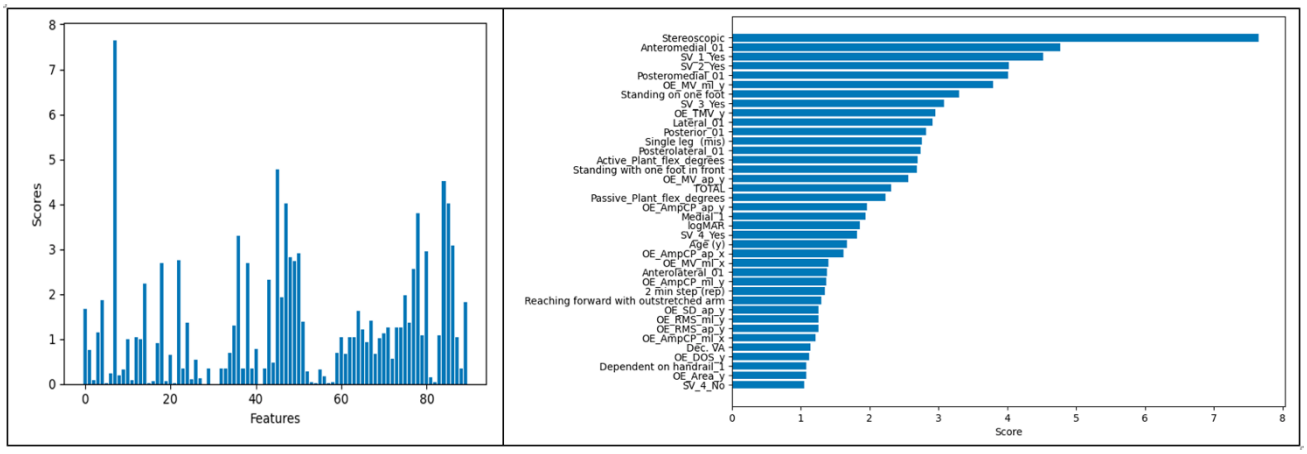


Figure 18. Feature Importance for Combined Non Stairs data (20% test split)(all - top 30 features)

4.2.7 OVER SAMPLING

Unlike CTGAN, SMOTE and random oversampling methods requires categorical features to be encoded so this step was performed later. Also this oversampling was performed for both non-stairs and stairs + non stairs dataset.

Both oversampling techniques were performed with minority sampling strategy to oversample minority class. Random state was set to avoid random sampling during each method call.

SMOTE synthesizes data based on neighbours of a randomly selected sample hence needs to know the number of neighbours - 'k_neighbors' parameter to define the neighbourhood to use, to generate samples. Almost all of the datasets used in this study used default of 5 neighbours, however for 30% split in both non-stairs and stairs + non stairs dataset, the decent fall input data neighbours were set to 4 as there were not sufficient minority class samples available.

4.2.8 FEATURE SELECTION

Further step was to select the features based on the importance score determined earlier. Here different approaches were used based on datasets.

For non-stairs data set, for each of the splits, for each of the fall category, the experiment was performed using (1) all the features as input and (2) top 30 features as input to the machine learning model. Once for each of the fall category, the best of these models with optimal parameters (determined after tuning) was determined, further impact of feature selection was determined, which is discussed in sections ahead.

In case of non-stairs + stairs dataset, feature selection was performed for 3 to 'n' (all) features i.e. for each fall category and each sampling strategy, the processed input with 3 features to n features was passed each time to the model. This was executed to get the best performing model, as only one model was used here with default parameters.

4.2.9 FEATURE SCALING

All the features are required to be on the same scale so that the machine learning model can better interpret the data. This is done by normalizing the range of features in the dataset. The numerical features were normalized by removing the mean and scaling to unit variance, before passing it to the model.

Once all these steps were completed, the data was ready to be passed on to the models. Considering the different train-test splits, number of features and the 3 fall categories, for non-stairs dataset as

whole, the total number of separate input datasets for each model were 24 (initial). While for stairs + non stairs data, the total input datasets were 9 * (n-3) for the model.

4.3 DIMENTIONALITY REDUCTION FOR VISUALIZATION

Before using the ML models, the original train data (20% split – Non Stairs data) was used with dimensionality reduction algorithms like Principal Component Analysis (PCA), t-distributed Stochastic Neighbor Embedding (t-SNE) and Uniform Manifold Approximation and Projection (UMAP), for the purpose of visualization in 2 dimensions. One key objective here was to check if the clusters formed after reducing the dimensions were separable enough to distinguish the falls so as, to check if it was useful enough for further modelling. For t-SNE, perplexity check was also done using KL Divergence metric as perplexity controls the effective number of neighbours that each point considers during the dimensionality reduction process.

4.4 MODELLING

Since this was a classification problem, classification based supervised machine learning algorithms were used. 5 models were tested for all the input datasets for non-stairs data whereas, just 1 model for non-stairs + stairs dataset. Since the experiment was first carried out for non-stairs dataset followed by stairs dataset, the models mentioned ahead are in order of execution for non-stairs data.

Initial plan was to use only combined fall data but the results were not optimal hence the need to check models with individual ascent as well as descent fall targets arose. This was necessary to analyse why models with combined falls as target variable behave in certain way.

4.4.1 MODEL TRAINING

First Logistic Regression with default parameters was first trained in hopes to investigate linear separability among classes.

Next was Logistic Regression with cross validation ('LogisticRegressionCV'). The method allowed use of Stratified k-fold cross validation which can help reduce overfitting. 5-fold cross validation was used for non-stairs dataset and 3-fold for non-stairs + stairs dataset. This was due to limited size of data even after oversampling especially with dataset 2. Additionally, Ridge (L2) penalty was also set to tackle overfitting of the model. Rest all parameter were set to default. This was the only model used with non-stairs + stairs dataset.

The third algorithm used was Support Vector Machine / Classifier, as it can handle non-linear decision boundaries by using kernel functions, and the dataset used didn't exhibit strong linear separability among classes. Hyperparameter tuning was performed using 'GridSearchCV' along with 5-fold cross

validation. The model was tuned for polynomial, linear and radial basis kernel, along with kernel specific parameters like 'gamma', and penalty parameter of error term 'C'.

Further, Tree based algorithms were explored. One of which was Random Forrest Classifier which is known to produce improved accuracy and reduce overfitting, as it uses multiple decision trees on various subsets of data. Additionally, it is robust to outliers and can handle non-linear data without using kernel functions. Similar to SVM, this model was executed with 5 fold cross validation while tuning the parameters. The model was tuned for number of decision trees and its maximum depth, minimum number of samples split at internal nodes or samples required at leaf nodes, etc in an effort to improve performance while reducing overfitting. The default Gini impurity criterion was applied.

The last model used was XGBoost with 5 fold cross validation and hyper tuning. Apart from being noted for better performance and high speed due to parallelized implementation of gradient boosting algorithm, XGBoost is capable of capturing complex patterns in data including non-linear relationships. The default booster 'gbtree' which, uses decision trees as base learners was used. The task being binary classification, the objective was set to 'binary:logistic'. Also, few tree based parameters like maximum depth, 'gamma' etc. were tuned.

For cross validation purpose, all models were evaluated using the 'roc_auc' score.

4.4.2 MODEL TESTING AND EVALUATION

Once trained, the models were tested for predicting if the subject has fallen or not for each fall/stair category with respective test sets. The model was evaluated using primarily using ROC-AUC rather than accuracy, as it takes into account true positive rate (TPR) and false positive rate (FPR) of the model across different thresholds making it robust to imbalanced data and it aligns well with the class imbalance in our test set. Also, next metric for model performance was Average Precision which is particularly useful when positive class is rare. Additionally other score like F1, accuracy, recall and precision were also determined. While predicting the outcome, Youden's J Statistic {max(TPR-FPR)} was applied to get optimal threshold probability, as it takes into account sensitivity and specificity making it robust to imbalance data. Additionally, while evaluating, the classification matrix was also determined to check model outputs (TP, TN, FP, FN).

4.5 DETERMINING BEST MODEL

Based on the evaluation metric, the best model was determined for respective datasets for each fall category, for each data split. These best models were specific to specific data sample (original/GANN/SMOTE/Random).

In case of Non-Stair Data, first with 20% split, model with all input features and top 30 input features was compared and best one determined. Similar with 30 % split. These initial best models with optimal tuned parameters were further trained (without further tuning) and tested for the same best

data sample by varying the number of features from minimum 3 to maximum ‘n’ (all) features based on importance (*feature impact*) to get the final best model for each category. While comparing the model ROC-AUC is taken into account. If same score, than average precision followed by F1 score.

In case of Stairs + Non Stairs Data, since we have only one model with default parameters, feature impact was directly applied and final best model was obtained.

Once the models were obtained, SHAP Analysis [42] was performed on best models to find how individual feature contributed to the output of the model.

The SHAP values are calculated by comparing prediction of a model with and without a feature, and the difference between the two prediction is the contribution of the feature to the final prediction. SHAP being additive in nature also helps in efficient computation even for high dimension datasets. Features with negative SHAP values negatively impact the predictions while those with positive values positively impact the model predictions.

Summary SHAP plots were plotted for each fall category to get a comprehensive a comprehensive view of most influential features. The results are discussed ahead.

4.6 EXPERIMENT SUMMARY (FLOW)

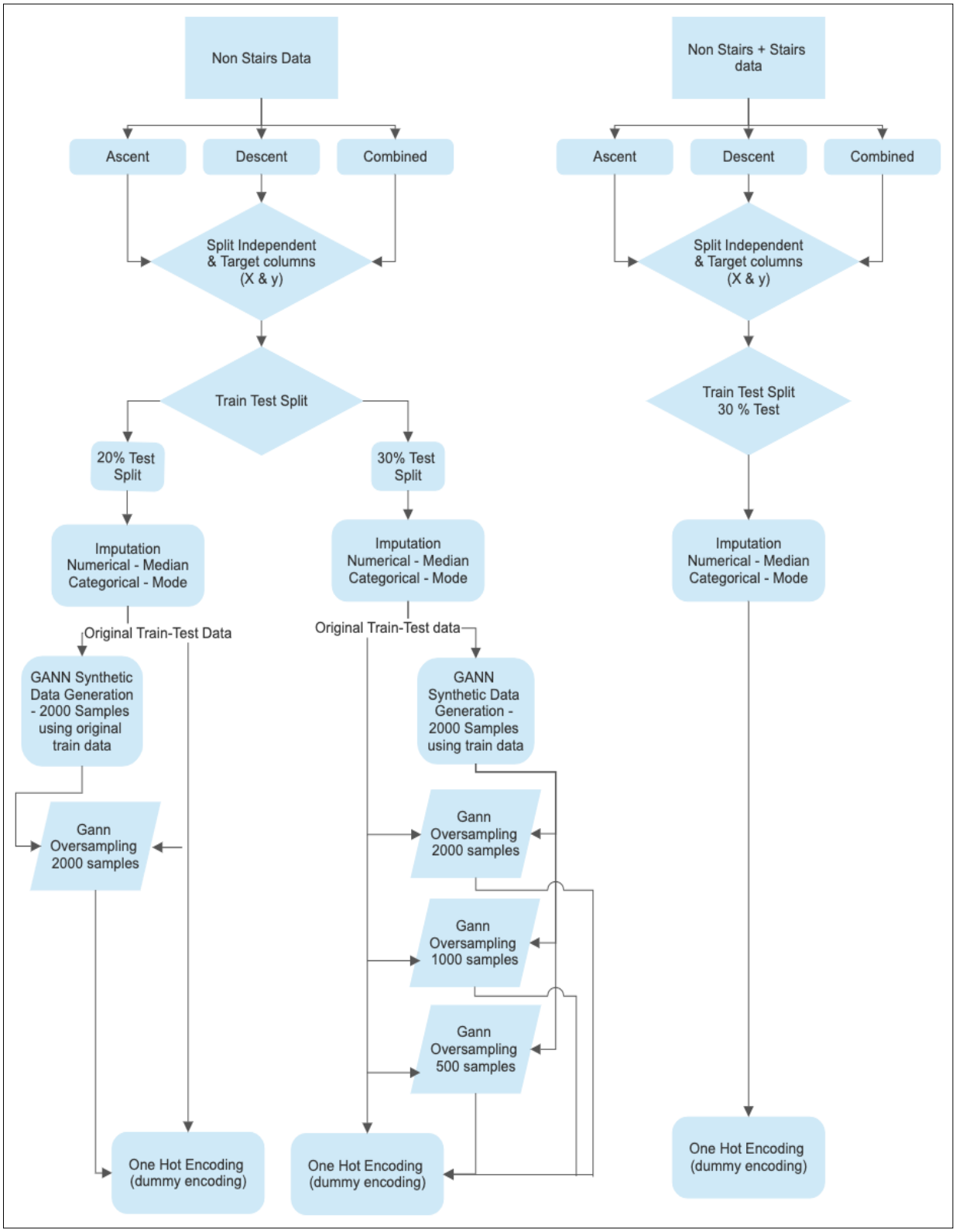


Figure 19. Detailed Experiment Flow - Part 1

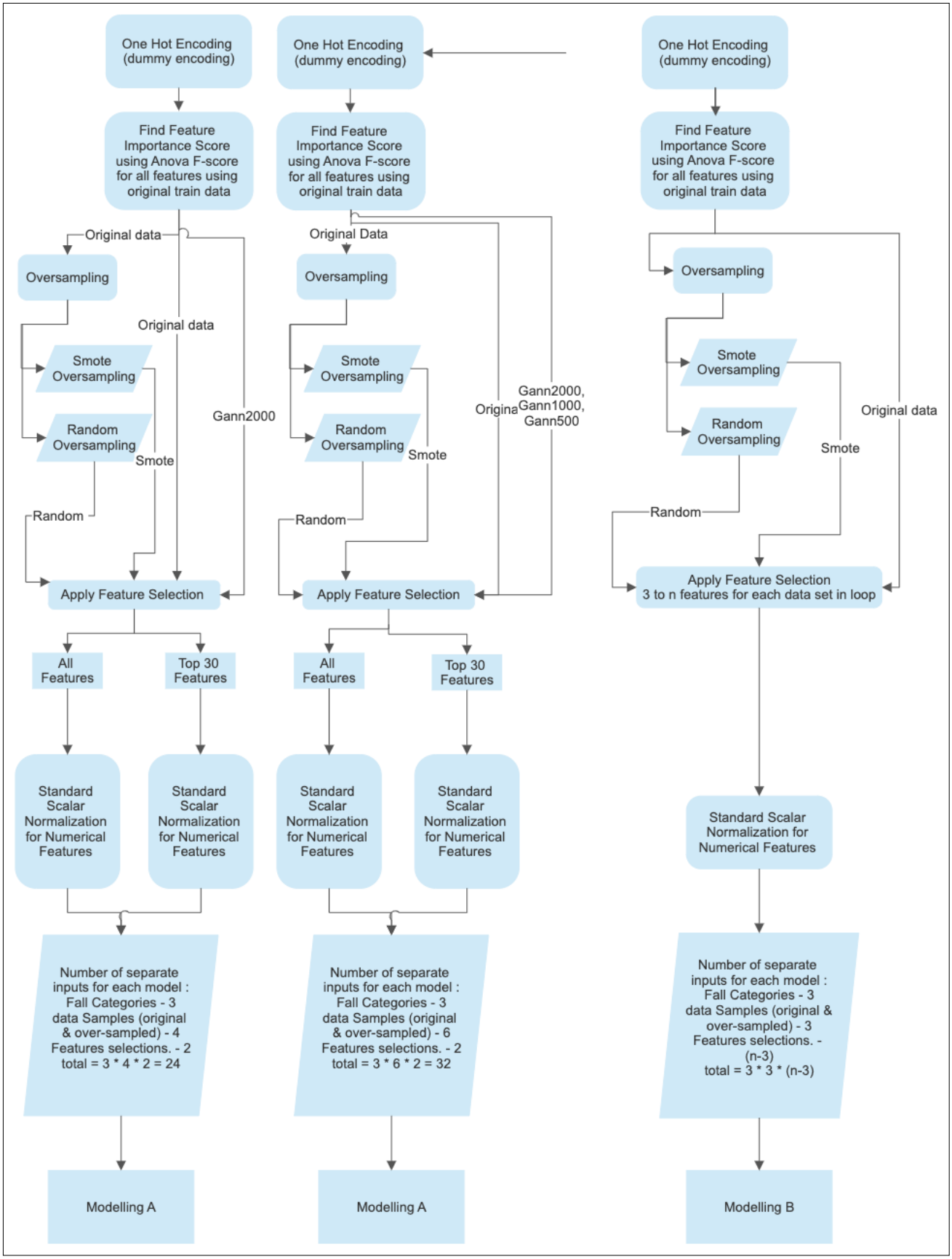


Figure 20. Detailed Experiment Flow - Part 2

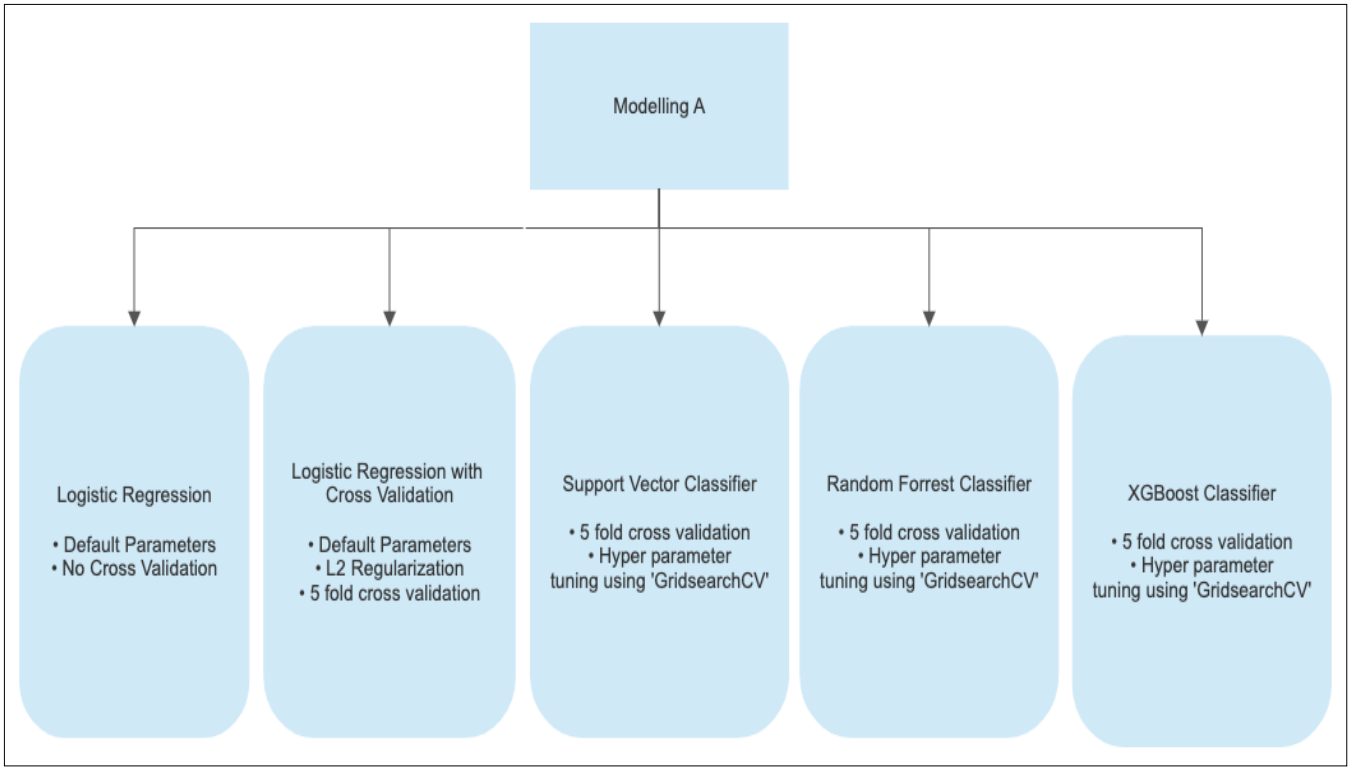


Figure 21. Detailed Experiment Flow - Part 3

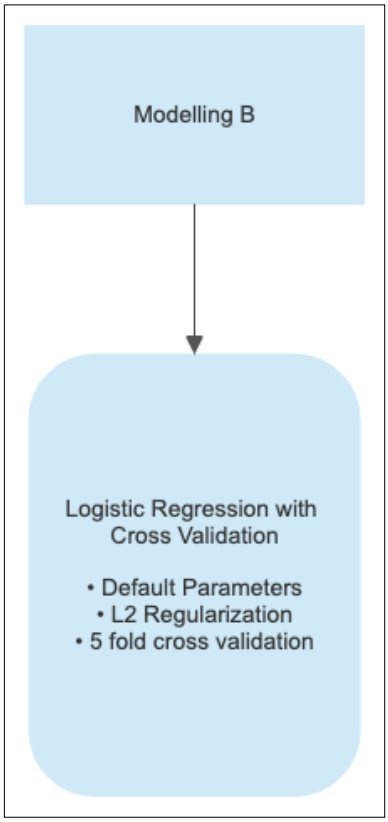


Figure 22. Detailed Experiment Flow - Part 4

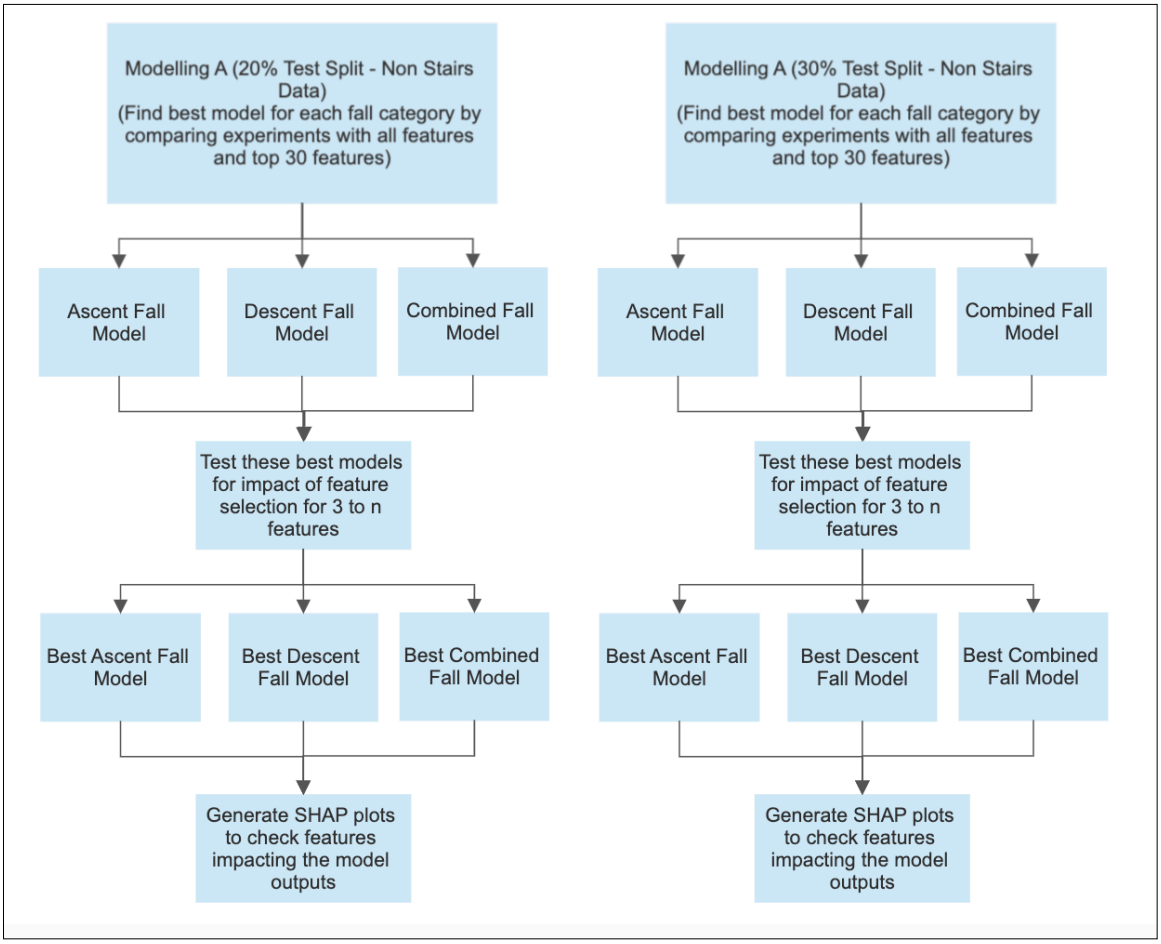


Figure 23. Detailed Experiment Flow - Part 5

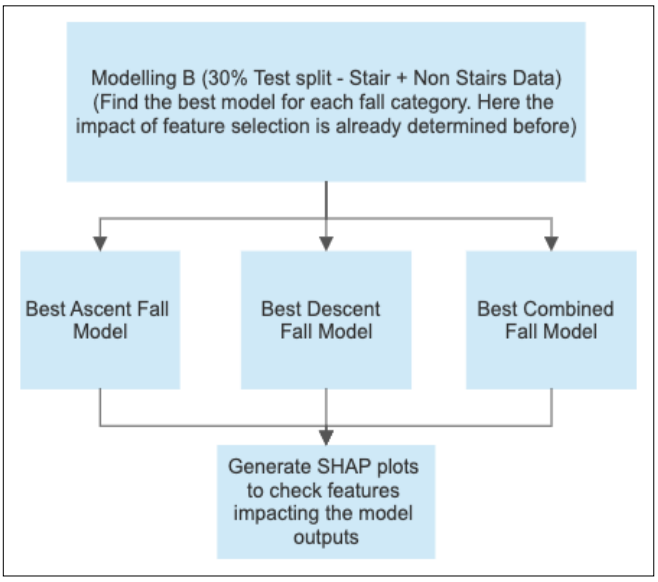


Figure 24. Detailed Experiment Flow - Part 6

5 RESULTS & DISCUSSION

5.1 DIMENSIONALITY REDUCTION FINDINGS

The classes were not linearly separable for either of the fall categories in either of the dimensionality reduction technique. No compact distinct clusters were visible with most closely coupled data points visible with PCA. Hence, these techniques were limited to just the original non-stairs data with all features for 20% test split and further dimensionality reduction experiments were not conducted with the other datasets.

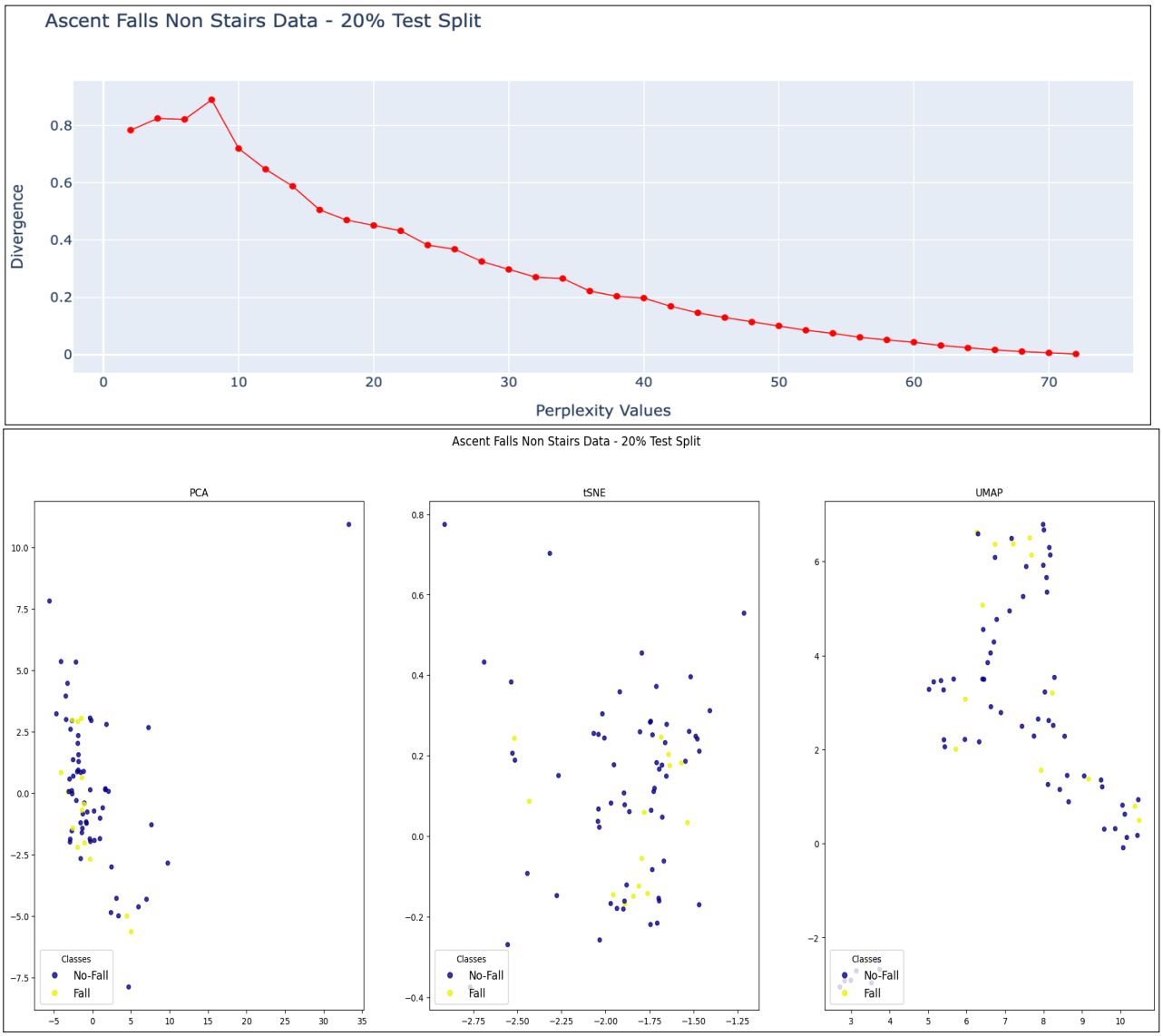


Figure 25. Non Stairs Data - Ascent Falls : Perplexity and Dimensionality Reduction

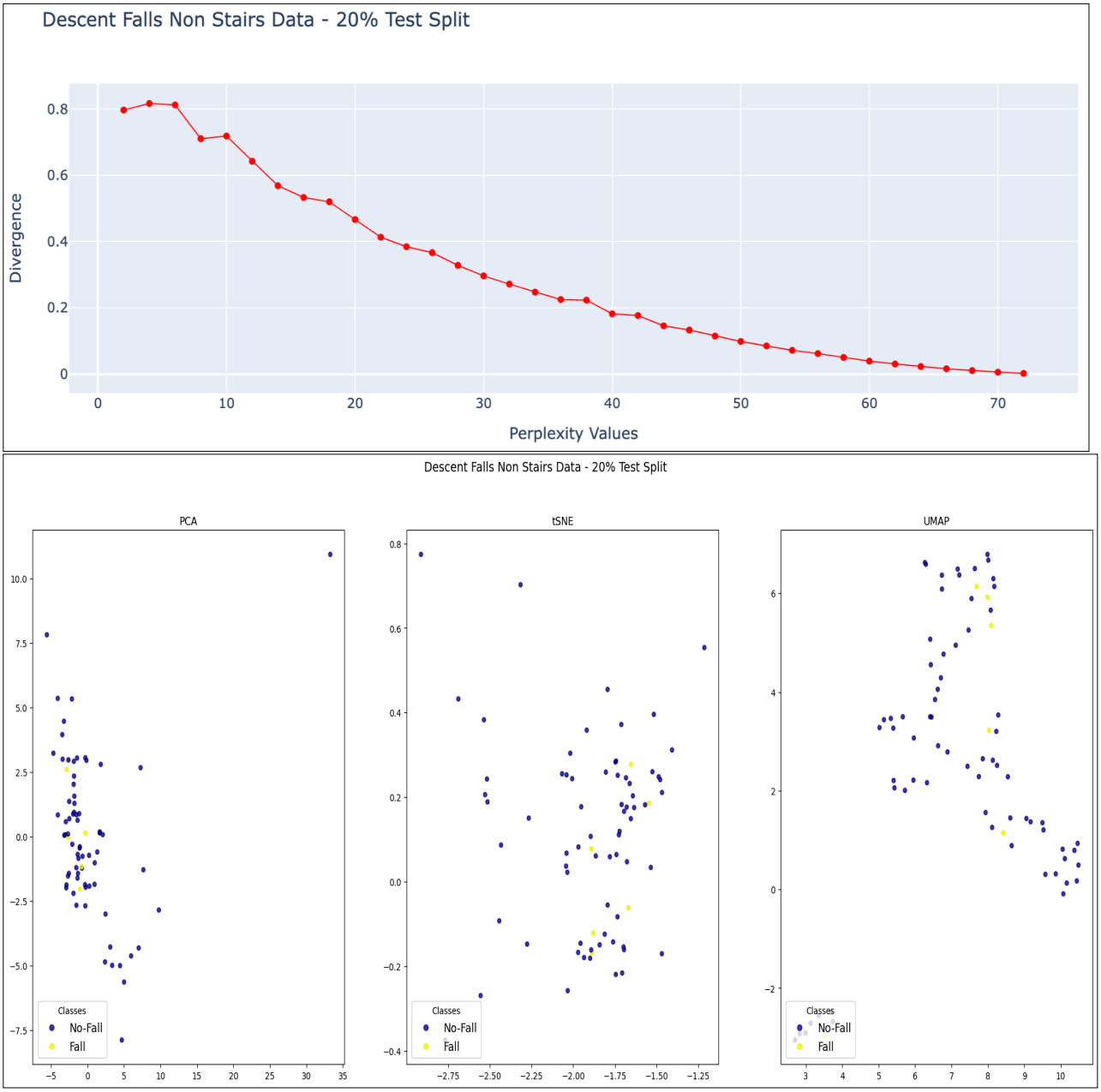


Figure 26. Non Stairs Data - Descent Falls : Perplexity and Dimensionality Reduction

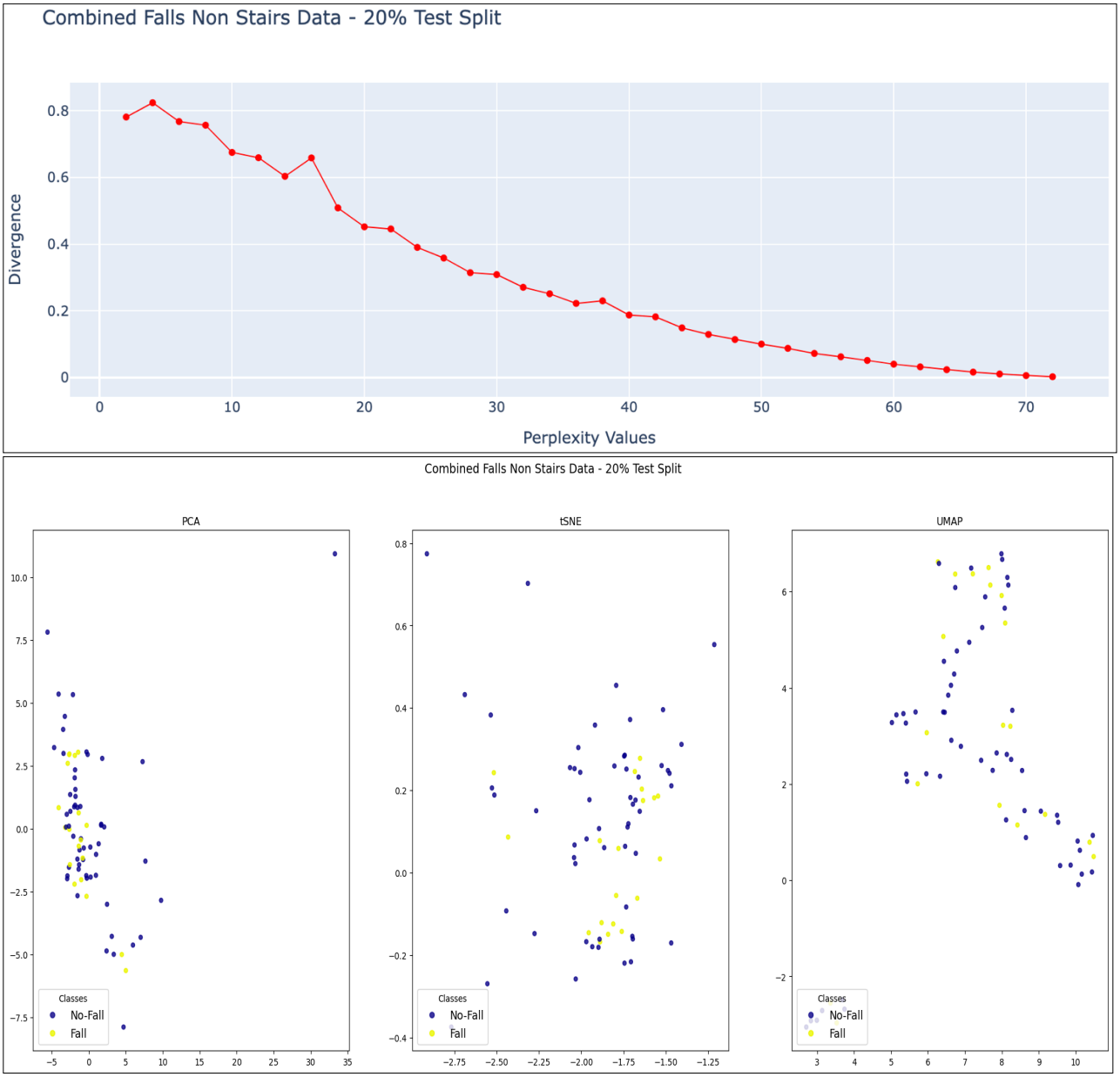


Figure 27. Non Stairs Data - Combined Falls : Perplexity and Dimensionality Reduction

5.2 GANN SYNTHETIC DATA QUALITY

Initially GANN based synthetic data was generated using 20% test split non stairs data (train data). With default synthesizer parameters, the quality score for 2000 samples was found to be below 70%. This may be due to the default parameters being on the higher end i.e. higher default values e.g. default batch size is 500 etc. CTGAN is typically used and performs well with abundant data and this analogy doesn't align with the train data being used here.

Reducing / Tuning the parameters improved the quality score significantly beyond 85% but couldn't go past beyond a threshold of 85% - 86%. CTGAN is based on neural networks and neural networks are known to perform well with significant amount of data which is not the case for this project and can be a likely reason for the low quality score.

```
Creating report:   0%|          | 0/4 [00:00<?, ?it/s]
Creating report: 100%|██████████| 4/4 [00:19<00:00,  4.87s/it]

Overall Quality Score: 85.82%

Properties:
Column Shapes: 89.13%
Column Pair Trends: 82.5%
Creating report: 100%|██████████| 4/4 [00:08<00:00,  2.05s/it]

DiagnosticResults:

SUCCESS:
✓ The synthetic data covers over 90% of the numerical ranges present in the real data
✓ The synthetic data covers over 90% of the categories present in the real data
✓ Over 90% of the synthetic rows are not copies of the real data
✓ The synthetic data follows over 90% of the min/max boundaries set by the real data
```

Figure 28. Quality Score and Diagnostic Report Output (20% Split)

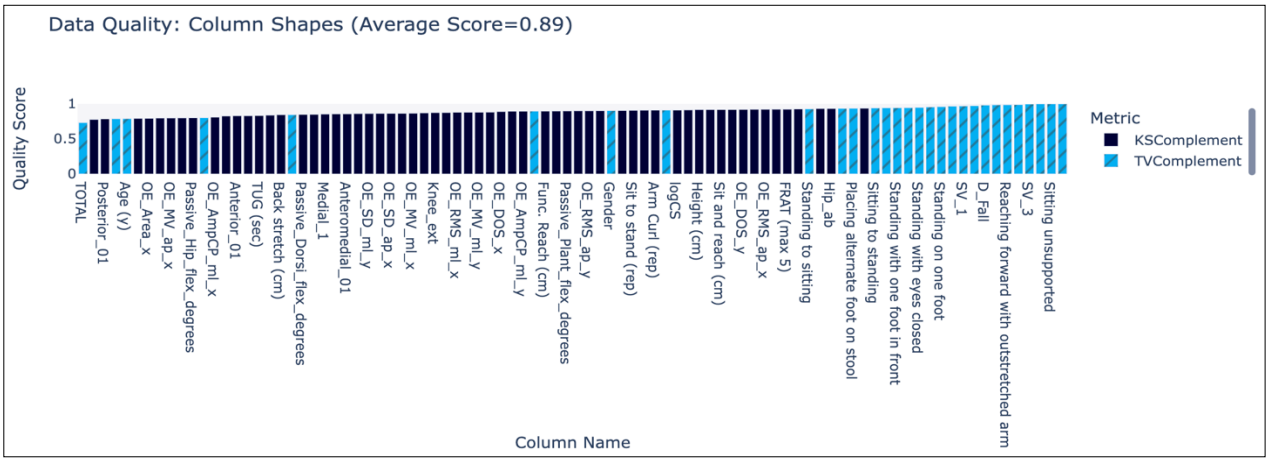


Figure 29. Column Shape Quality (20% test split data)

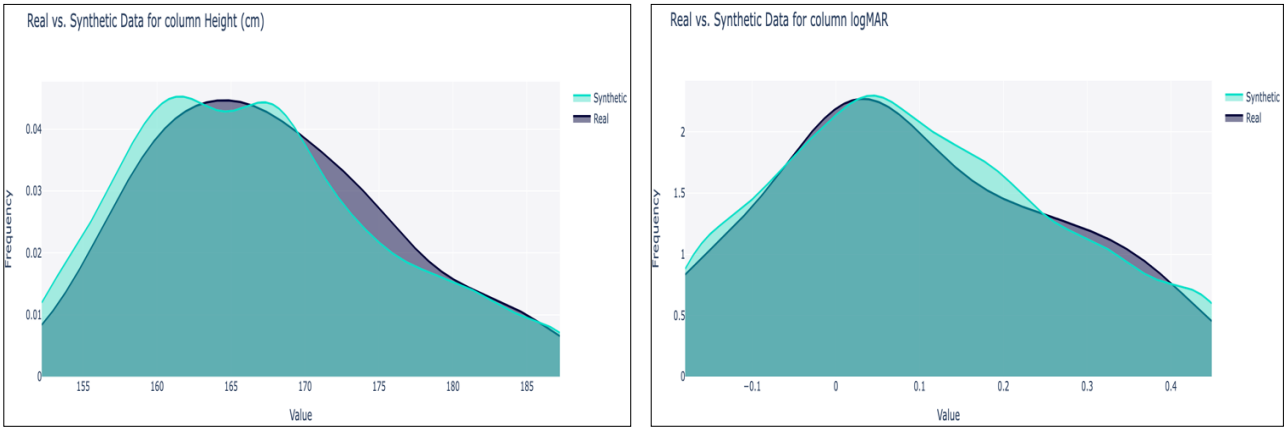


Figure 30. Numerical Feature Distributions – Height and logMAR (Real vs Synthetic data) (20% split)

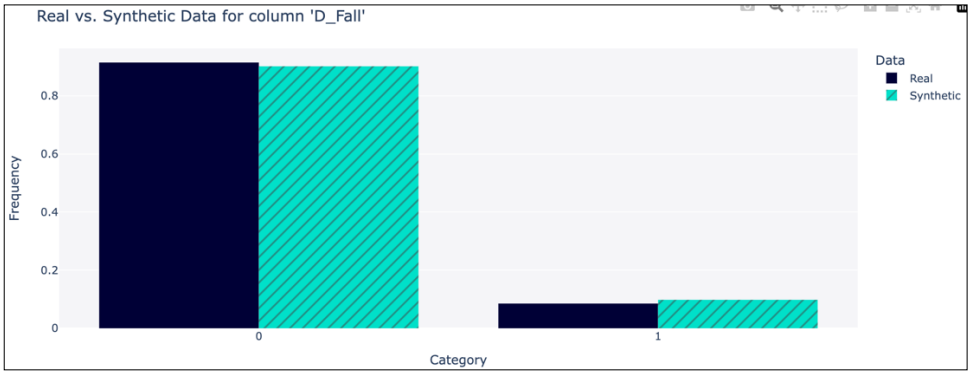


Figure 31. Categorical Feature Distribution - Descent Falls (Real vs Synthetic data) (20% split)

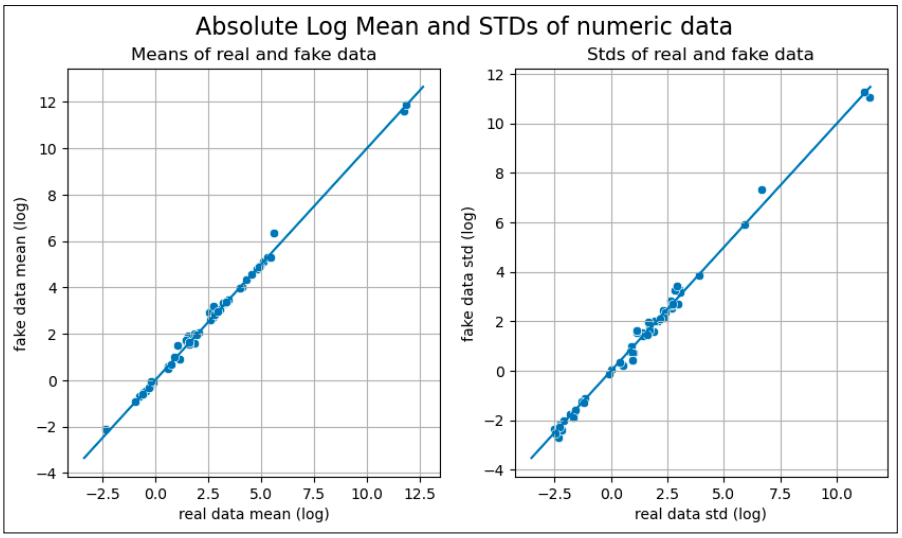


Figure 32. Mean and Standard Deviation Trend (Real vs Synthetic Data) (20% split)

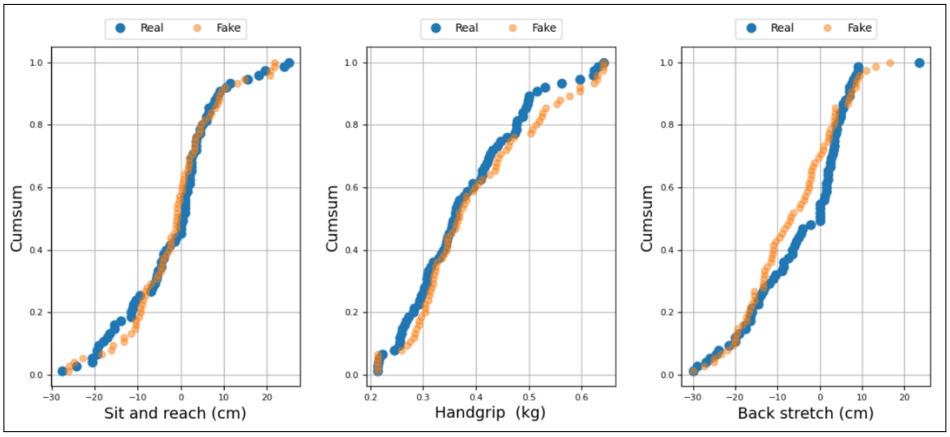


Figure 33. Real vs Fake Cumulative Sum(20% split)

For synthetic data generated using 30% test split, similar quality score was observed with 2000 samples however, as less samples were generated (1000, 500) the quality started to slightly diminish as seen below.

2000 Synthetic Samples

```
Creating report: 100%|██████████| 4/4 [00:21<00:00,  5.27s/it]

Overall Quality Score: 85.73%

Properties:
Column Shapes: 89.13%
Column Pair Trends: 82.33%
Creating report: 100%|██████████| 4/4 [00:09<00:00,  2.49s/it]

DiagnosticResults:

SUCCESS:
✓ The synthetic data covers over 90% of the numerical ranges present in the real data
✓ The synthetic data covers over 90% of the categories present in the real data
✓ Over 90% of the synthetic rows are not copies of the real data
✓ The synthetic data follows over 90% of the min/max boundaries set by the real data
```

1000 Synthetic Samples

```
Creating report: 100%|██████████| 4/4 [00:20<00:00,  5.05s/it]

Overall Quality Score: 85.55%

Properties:
Column Shapes: 88.97%
Column Pair Trends: 82.13%
Creating report: 100%|██████████| 4/4 [00:09<00:00,  2.48s/it]

DiagnosticResults:

SUCCESS:
✓ The synthetic data covers over 90% of the numerical ranges present in the real data
✓ The synthetic data covers over 90% of the categories present in the real data
✓ Over 90% of the synthetic rows are not copies of the real data
✓ The synthetic data follows over 90% of the min/max boundaries set by the real data
```

500 Synthetic Sample

```
Creating report: 100%|██████████| 4/4 [00:19<00:00,  4.77s/it]

Overall Quality Score: 85.31%

Properties:
Column Shapes: 88.68%
Column Pair Trends: 81.93%
Creating report: 100%|██████████| 4/4 [00:10<00:00,  2.56s/it]

DiagnosticResults:

SUCCESS:
✓ The synthetic data covers over 90% of the numerical ranges present in the real data
✓ The synthetic data covers over 90% of the categories present in the real data
✓ Over 90% of the synthetic rows are not copies of the real data
✓ The synthetic data follows over 90% of the min/max boundaries set by the real data
```

Figure 34. Quality Score and Diagnostic Report (30% split) - Reduction in score as sample size decrease

5.3 DATA SHAPE

As oversampling was used, the instances generated in train data from each oversampling technique varied significantly, with GANN producing the maximum instances. Test data was limited with very few instanced in 20% split as expected.

Table 5. Non Stairs Data Shape - 20% split

Non Stairs Data (20% split)				
Fall Category	**Data Sample**	**Train(rows)**	**Max Features (cols)**	**Test (rows)**
Ascent	Original	75	90	19
	Smote	122		
	Random	122		
	GANN (2000)	642		
Descent	Original	75	90	19
	Smote	138		
	Random	138		
	GANN (2000)	404		
Combined	Original	75	90	19
	Smote	112		
	Random	112		
	GANN (2000)	986		

Table 6. Non Stairs Data Shape - 30 % split

Non Stairs Data (30% split)				
Fall Category	**Data Sample**	**Train (rows)**	**Max Features (cols)**	**Test (rows)**
Ascent	Original	65	90	29
	Smote	106		
	Random	106		
	GANN2000	518		
	GANN1000	246		
	GANN500	140		
Descent	Original	65	90	29
	Smote	120		
	Random	120		
	GANN2000	320		
	GANN1000	186		
	GANN500	94		
Combined	Original	65	90	29
	Smote	98		
	Random	98		
	GANN2000	812		
	GANN1000	416		
	GANN500	226		

Similar pattern was observed with non-stairs + stairs data set in terms of proportion however significantly less instances were available as this data only had subjects non-dependant on hand rail.

5.4 MODEL EVALUATION AND FEATURE IMPACT

5.4.1 NON STAIRS DATA – 20% SPLIT

The test was first initiated with Non Stairs Data with 20% test split. This was further executed in 2 phases. First, experiment 1 – Modelling with all input features for respective data set. However, the results not being optimal, second phase, experiment 2 – Modelling with top 30 features was initiated to improve the predicting outcome.

5.4.1.1 EXPERIMENT 1 - MODEL PERFORMANCE WITH ALL FEATURES

1) Combined Fall Data
   a) Best result obtained via XGBoost using GANN sampled train data with AUC - 0.77 and AP - 0.60. *('_CV' as suffix to model names represent model with cross validation applied)*
   b) Here original data and Gann sample data performed well with different models compared to the other oversampled datasets.

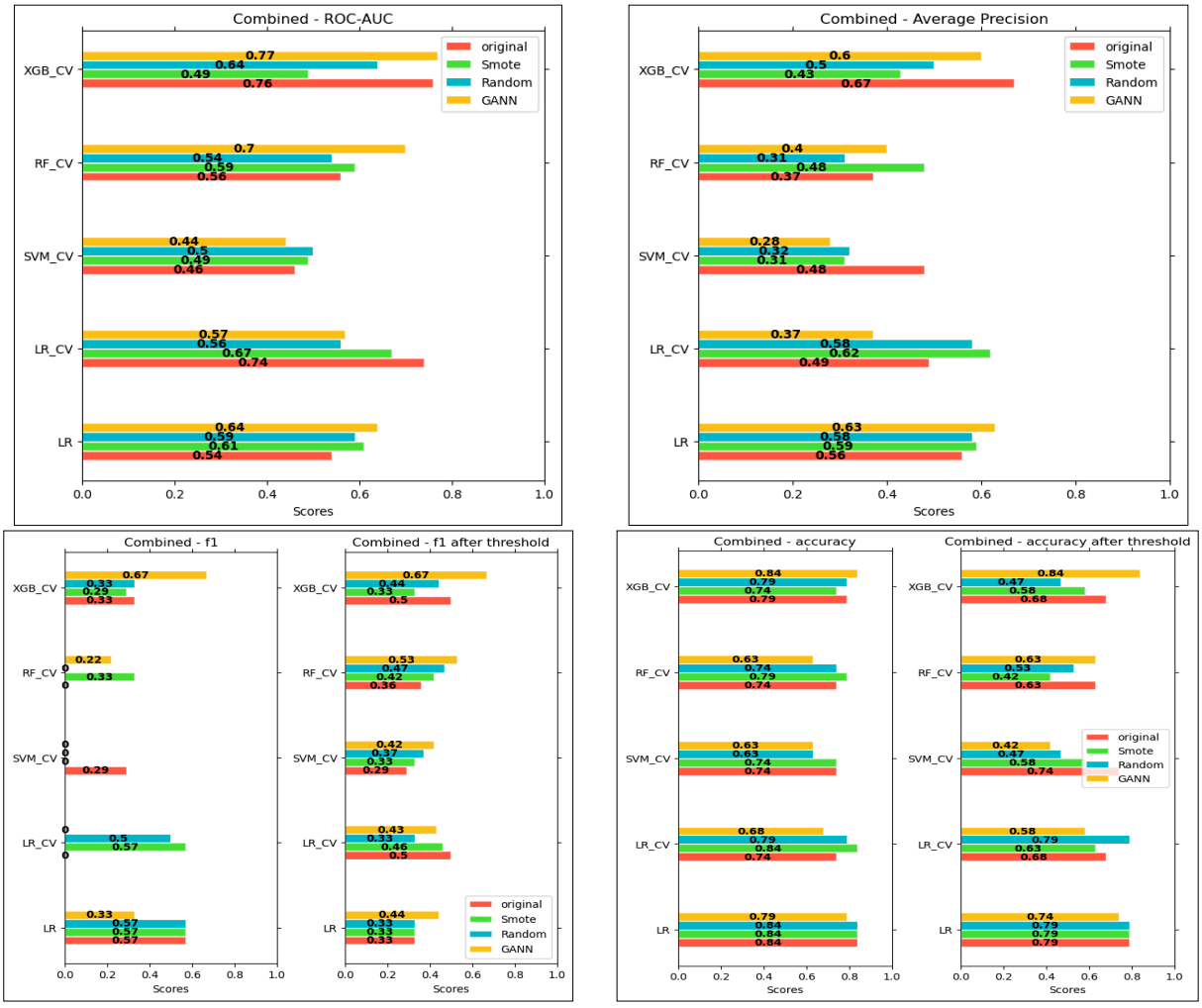


Figure 35. Model Scores - Combined Falls - Non Stairs Data (20% split)

2) Ascent Fall Data

a) Best result obtained via Logistic regression with cross validation using original train data with AUC - 0.77 and AP - 0.45.

b) None of the over sampled data produced AUC score better than 0.65 apart from GANN data which was around 0.73.

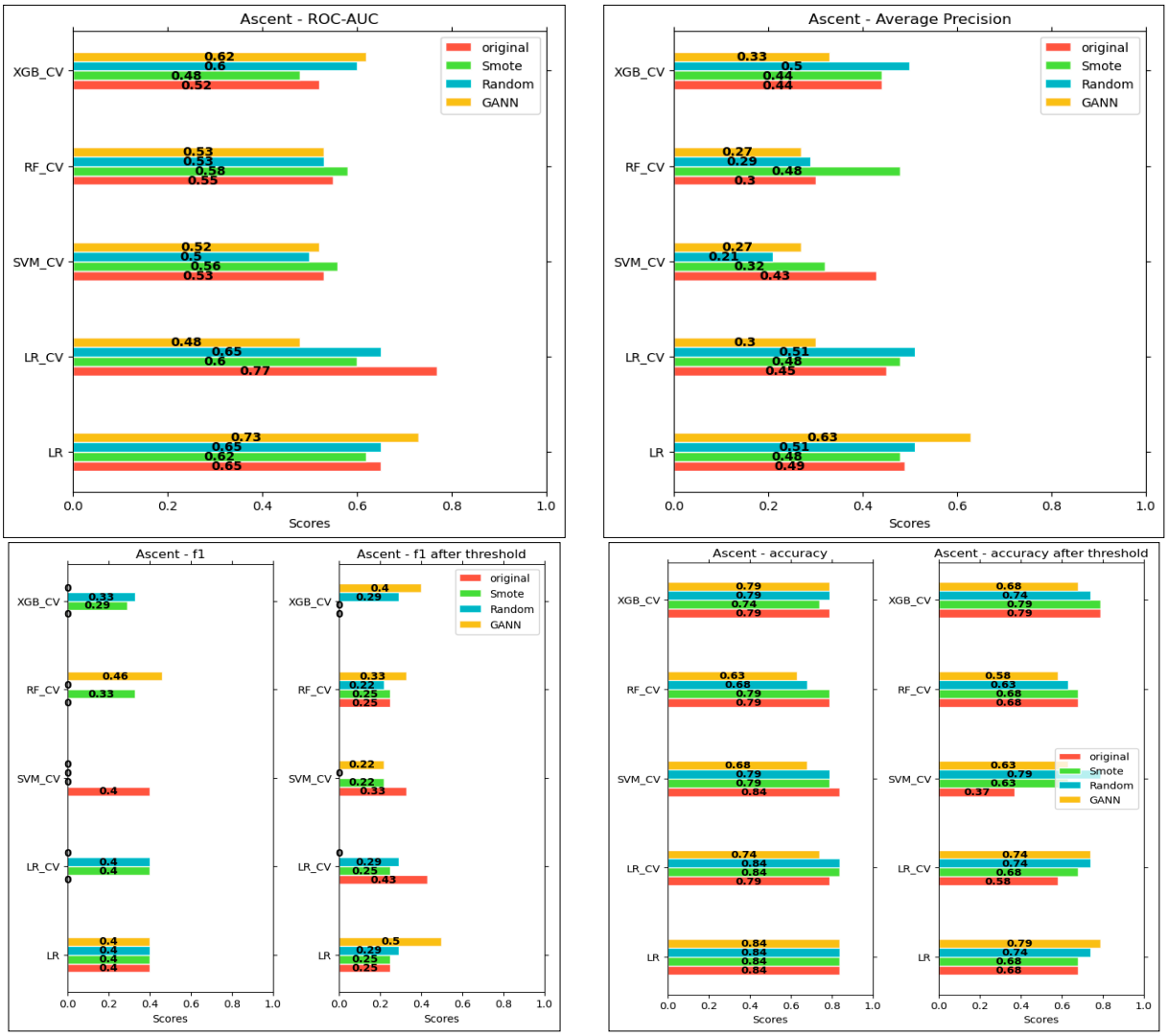


Figure 36. Model Scores - Ascent Falls - Non Stairs Data (20% split)

3) Descent Fall Data
   a) Best result obtained via Logistic regression with cross validation using GANN sampled train data with AUC - 0.82 and average precision of 0.39.
   b) Unlike as seen with Ascent data, oversampled descent data produced better results and original data particularly no so good.

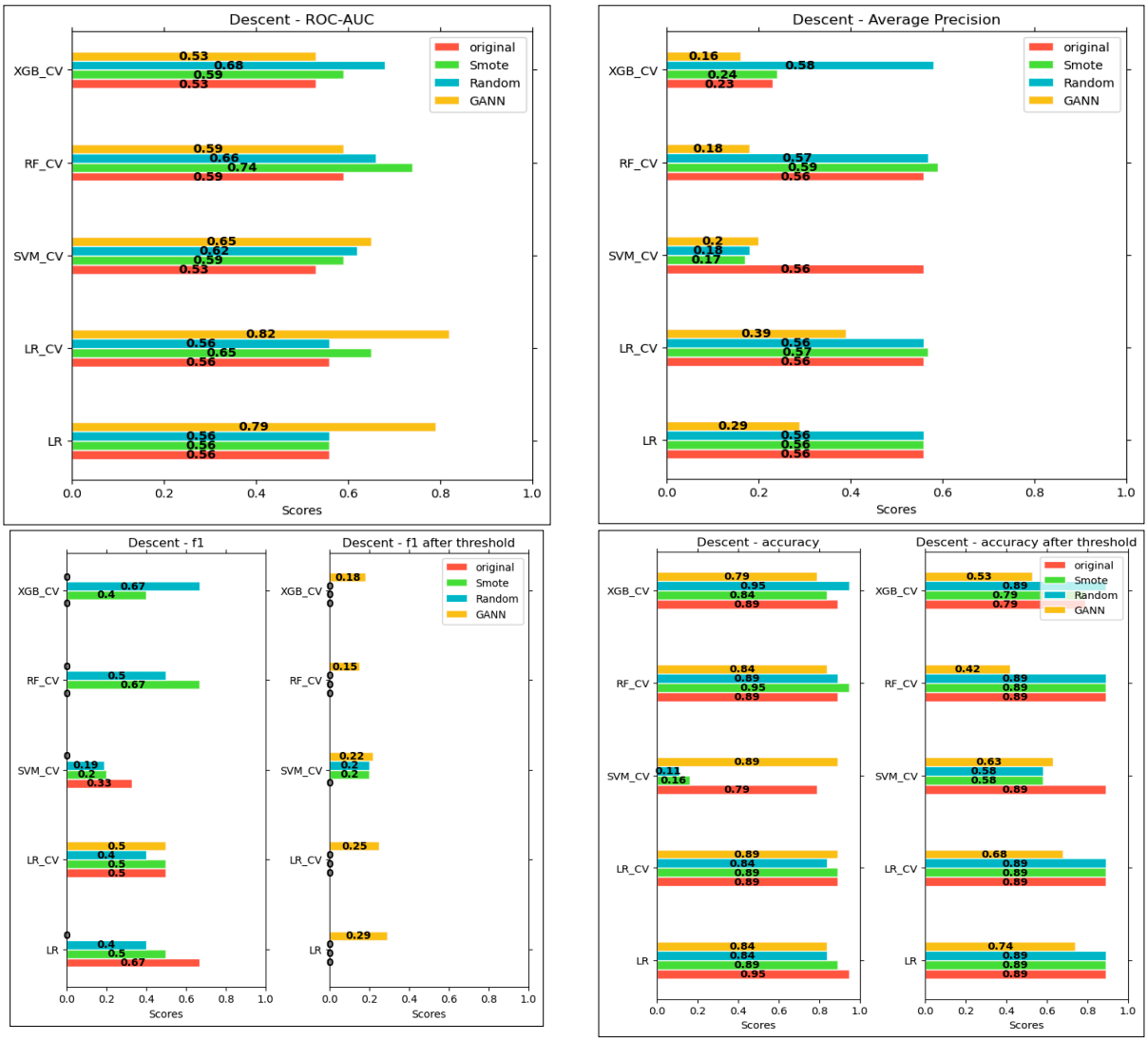


Figure 37. Model Scores - Combined Falls - Non Stairs Data (20% split)

The performance was not optimal (AUC < 0.85) for either of the fall categories with either of the models, with either of the data samples when all the features were utilised. After analysing the three models for then 3 falls, we can say descent fall prediction comparatively performed well. The reason for performance for combined fall prediction to be on the lower end is likely because of ascent data (as combined fall is combination of ascent and descent) and ascent fall prediction was also not ideal. This analogy can be related w.r.t the model as well. As ascent predictions were best using logistic regression with cross validation and original ascent fall data, we can observe that for combined predictions, the same model with original combined fall data performed third best considering AUC.

5.4.1.2 EXPERIMENT 2 - MODEL PERFORMANCE WITH TOP 30 FEATURES

Since the models didn't perform well the data with all features was utilized, the next experiment was to check with top 30 features which accounted to one-third of total input data features. The results significantly improved as is discussed as follows.

1) Combined Fall Data
   a) Best result obtained via XGBoost using GANN sampled train data with AUC - 0.80 and average precision of 0.66.
   b) Apart from Random forest classifier, all other models produced better result with one or other dataset, with AUC >= 0.70 if compared to earlier case.

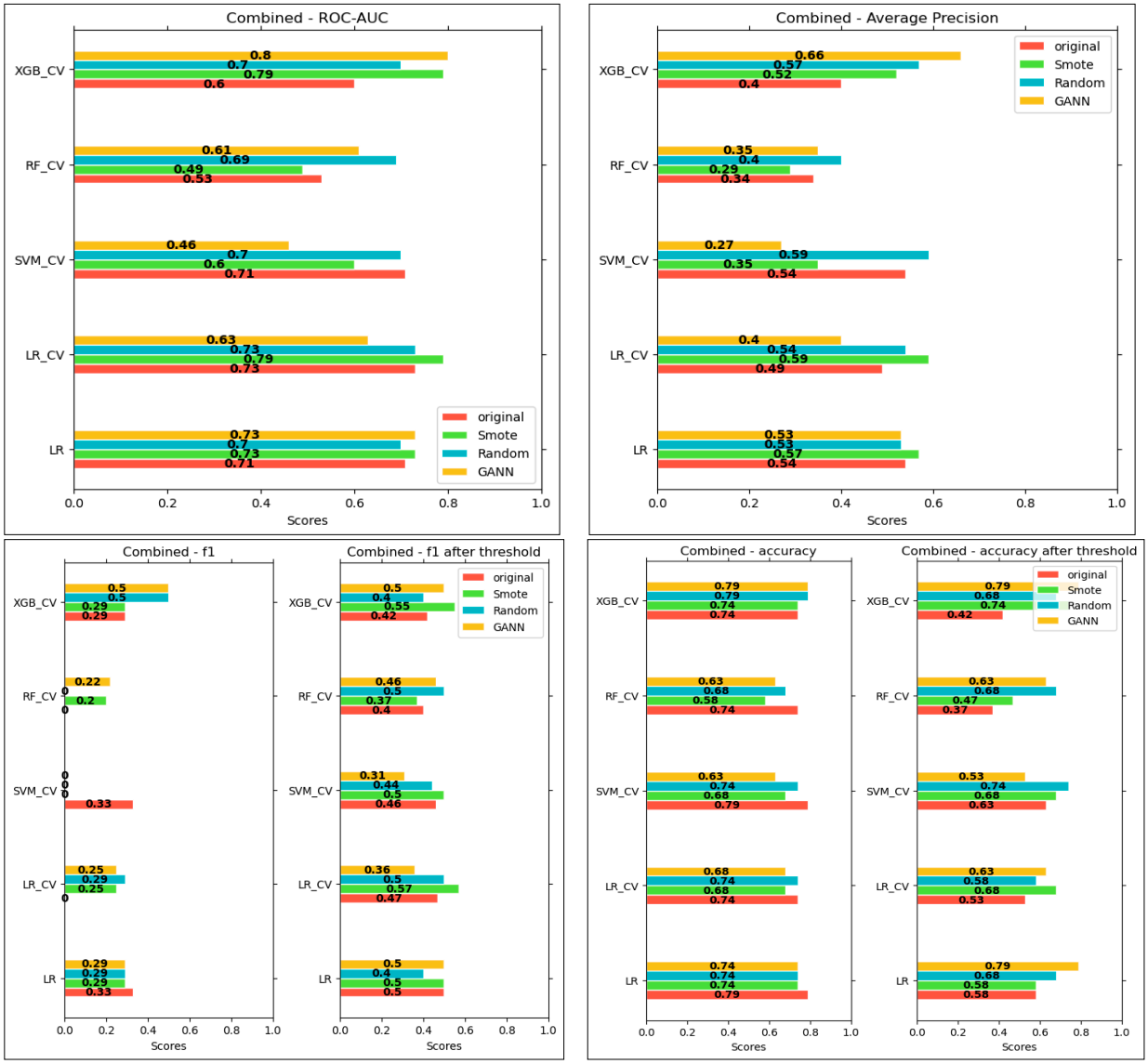


Figure 38. Model Scores - Combined Falls - Non Stairs Data (20% split, Top 30 Features)

Similar to previous case, with objective of improving the results, the experiment was conducted for the other two fall categories.

2) Ascent Fall Data

a) Best result obtained via XGBoost using GANN sampled train data with AUC - 0.90 and average precision of 0.69.

b) As seen in previous case, original data with logistic regression (cross validation) also proved one of the better results. Overall, significant improvement was observed.

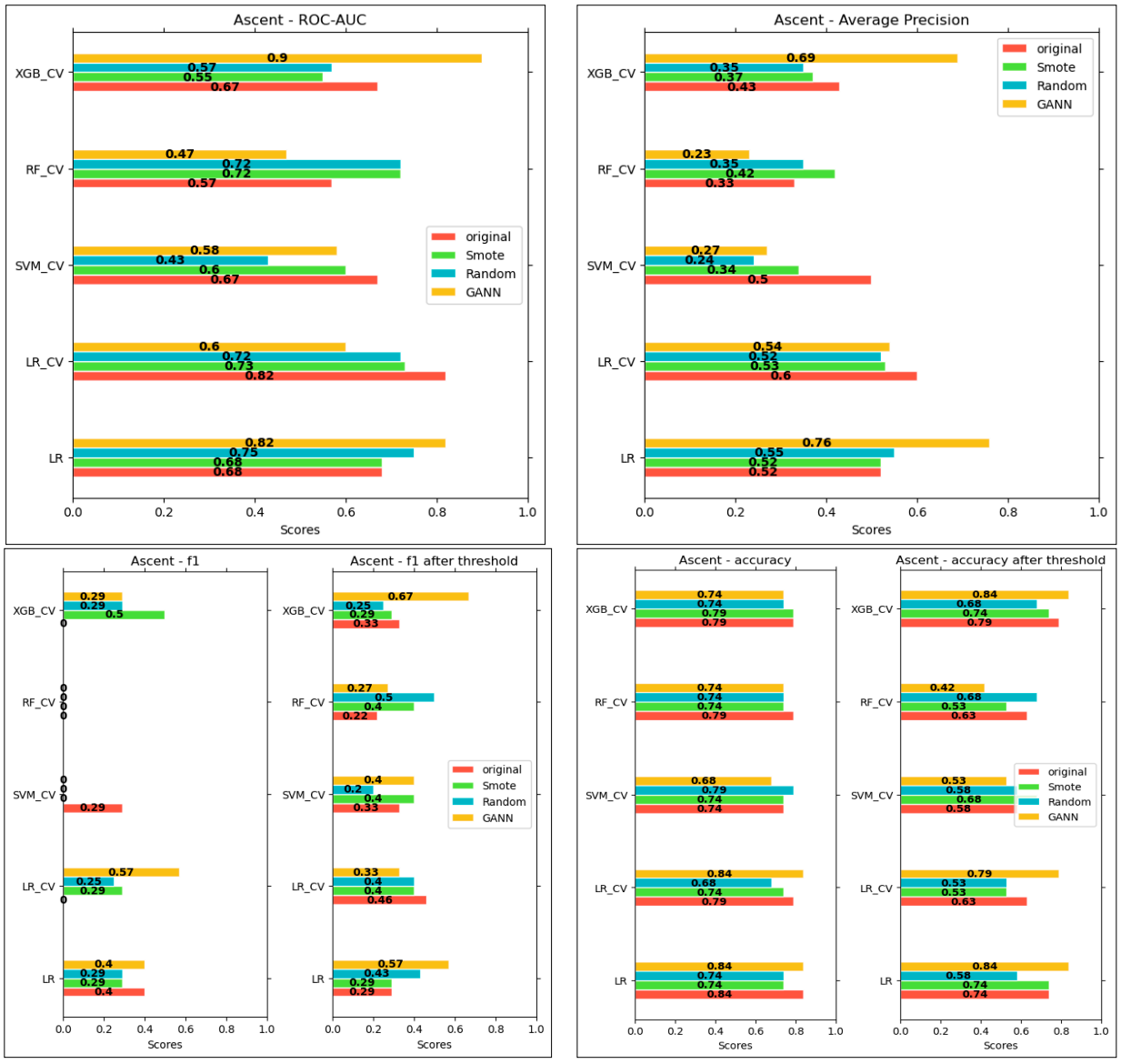


Figure 39. Model Scores - Ascent Falls - Non Stairs Data (20% split, Top 30 Features)

3) Decent Fall Data

a) Best result obtained via Logistic Regression with Cross validation using GANN sampled train data with AUC - 0.85 and average precision of 0.42.

b) Apart from the best score, similar AUC score patters were observed when compared to previous case.

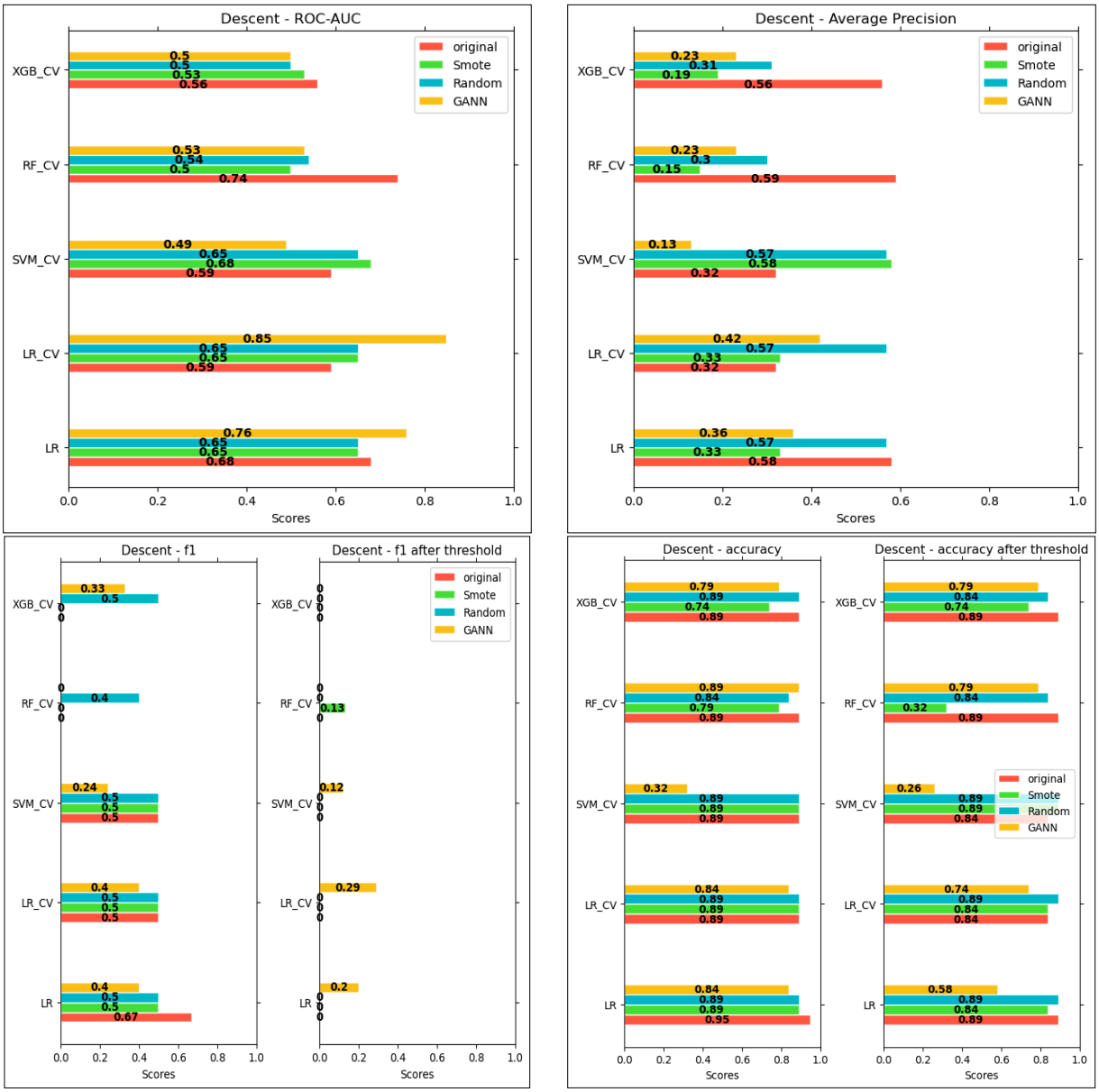


Figure 40. Model Scores - Descent Falls - Non Stairs Data (20% split, Top 30 Features)

5.4.1.3 COMPARING THE TWO EXPERIMENTS

Overall significant improvement in prediction results, especially for ascent falls was observed when only top 30 features were used. Additionally, GANN oversampled data performed significantly well for the best models in both the experiments for all the fall categories except Ascent fall in Experiment 1 which produced best result with original data. Also, GANN sampled data being more in size compared to the other oversampled data could also have played a part in model performances as general rule in ML says the more quality data we have the more our model will learn about the data, and better will be the outcome.

Table 7. Best Model Comparisons - All vs Top 30 Features (Non Stairs Data - 20% split)

	All Features				Top 30 Features			
Stairs	**Data**	**Model**	**ROC-AUC**	**AP**	**Data**	**Model**	**ROC-AUC**	**AP**
Ascent	Original	LR_CV	0.77	0.45	GANN	XGB_CV	0.90	0.69
Descent	GANN	LR_CV	0.82	0.39	GANN	LR_CV	0.85	0.42
Combined	GANN	XGB_CV	0.77	0.60	GANN	XGB_CV	0.80	0.66

Logistic Regression with cross validation performed well for descent in both the cases using the GANN data. Underlying reasons for this could likely to be due discovering better linear relationships between the input features and target – descent fall when compared to ascent or combined and, less noisy features.

The best models for ascent and combined falls were XGBoost based and this may likely be due the fact that XGBoost iteratively adjusts its parameters, sequentially creating the decision trees with each tree improving upon the mistake of the previous ones, allowing for learning complex patterns within the data.

Combined falls alone for fall prediction as a whole was not found to be useful as compared to determining ascent and descent falls individually, which is evident from the performance score (AUC) of >=0.85 observed when fall predictions are individually computed.

5.4.1.4 CHECKING FEATURE IMPACT ON THE BEST MODELS

Once the initial best models (with top 30 features) with optimal parameters were determines, they were further trained and tested with varying the number of features based on importance. This effort was found out to be productive as the results improved significantly.

ASCENT FALL DATA MODEL – XGB & GANN (GAN SAMPLED DATA FROM 2000 SYNTHETIC SAMPLES)

The model produces same scores : ROC-AUC = 0.90 and AP = 0.69, however the minimum features required to produce best results reduced to 27 from initial 30 (Figure 41). Best final model performance was obtained after applying the threshold .

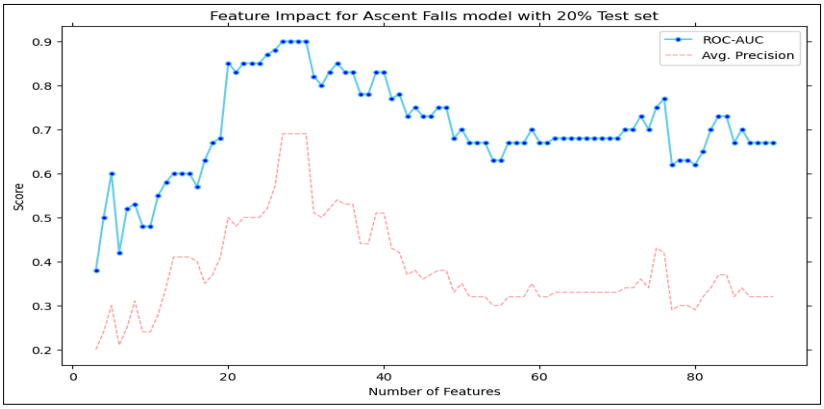


Figure 41. Feature Impact on Best Ascent Falls Model (XGBoost, 20% Test Split, Non Stairs Data, GANN Sampled)

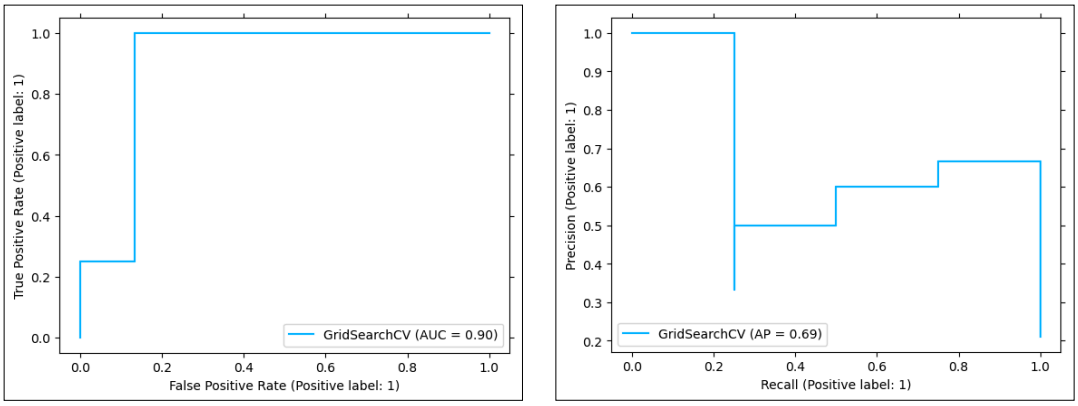


Figure 42. ROC (left) and Precision Recall (right) Curves – Final Best Ascent Falls Model

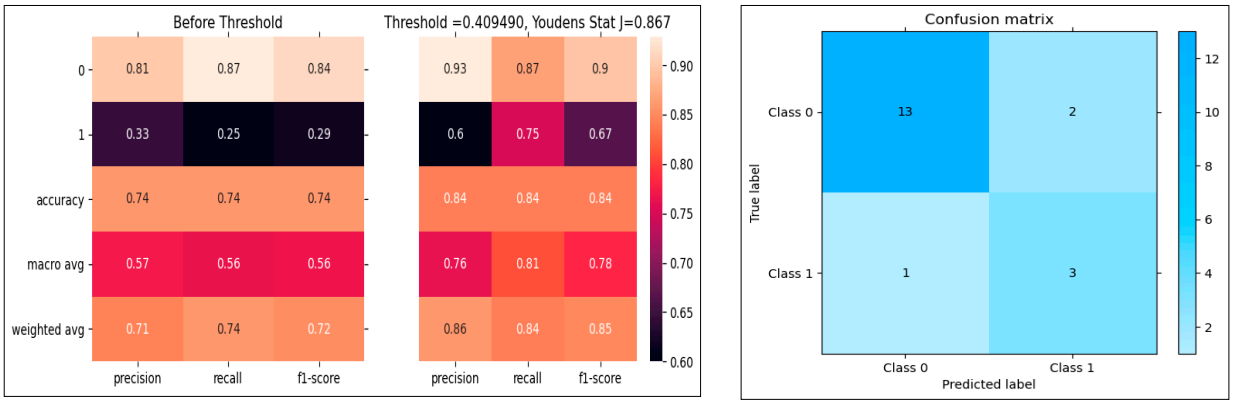


Figure 43. Classification Report (left) and Classification Matrix (right) – Final Best Ascent Falls Model

DESCENT FALL DATA MODEL – LRCV & GANN (GAN SAMPLED DATA FROM 2000 SYNTHETIC SAMPLES)

The model scores significantly improved : ROC-AUC = 0.94 and AP = 0.58, however the minimum features required to produce best results increased to 67 from initial 30 (Figure 44). Best final model performance was obtained after applying the threshold .

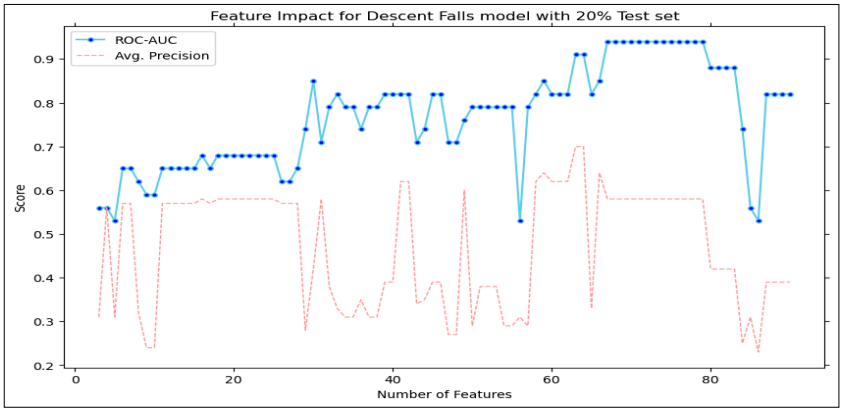


Figure 44. Feature Impact on Best Descent Falls Model (Logistic Regression with cross validation, 20% Test Split, Non Stairs Data, GANN Sampled)

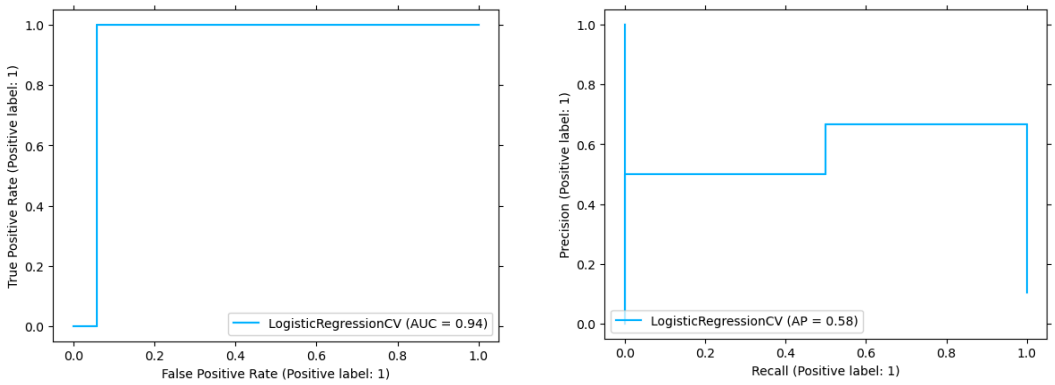


Figure 45. ROC (left) and Precision Recall (right) Curves – Final Best Descent Falls Model

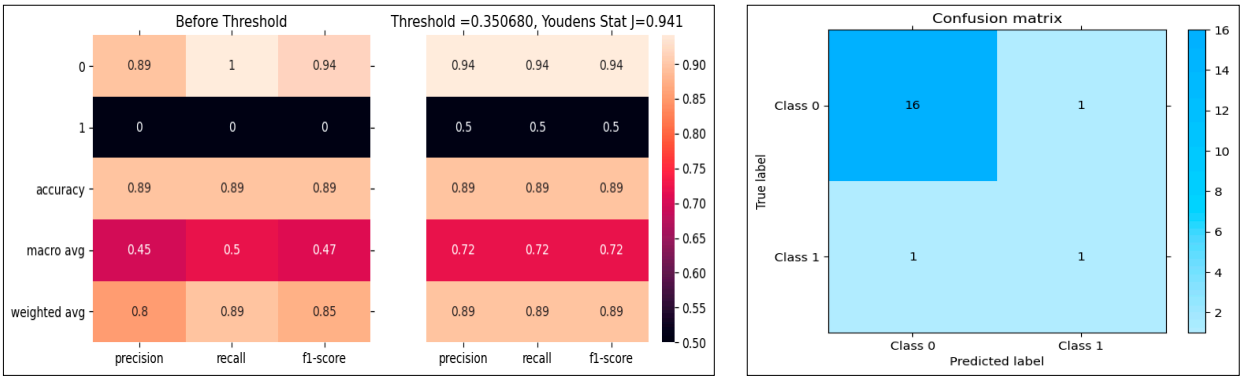


Figure 46. Classification Report (left) and Classification Matrix (right) – Final Best Descent Falls Model

COMBINED FALL DATA MODEL – XGB & GANN (GAN SAMPLED DATA FROM 2000 SAMPLES)

The model score slightly improved : ROC-AUC = 0.84 and AP = 0.65, and the minimum features required to produce best results decreased to 19 from initial 30 (Figure 4). Best final model performance was obtained after applying the threshold .

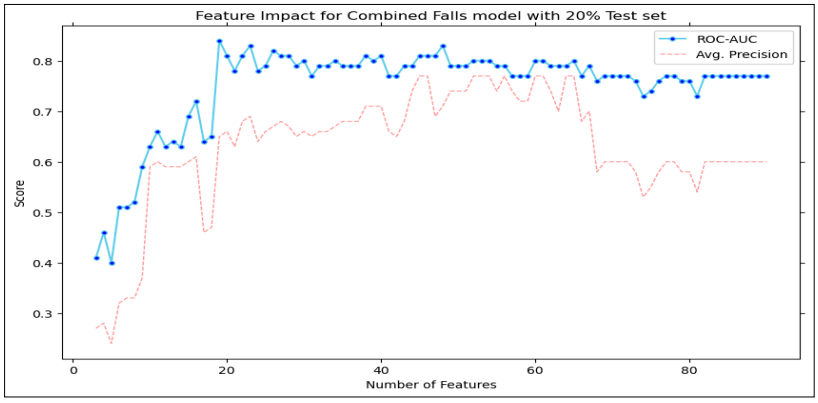


Figure 47. Feature Impact on Best Combined Falls Model (XGBoost, 20% Test Split, Non Stairs Data, GANN Sampled)

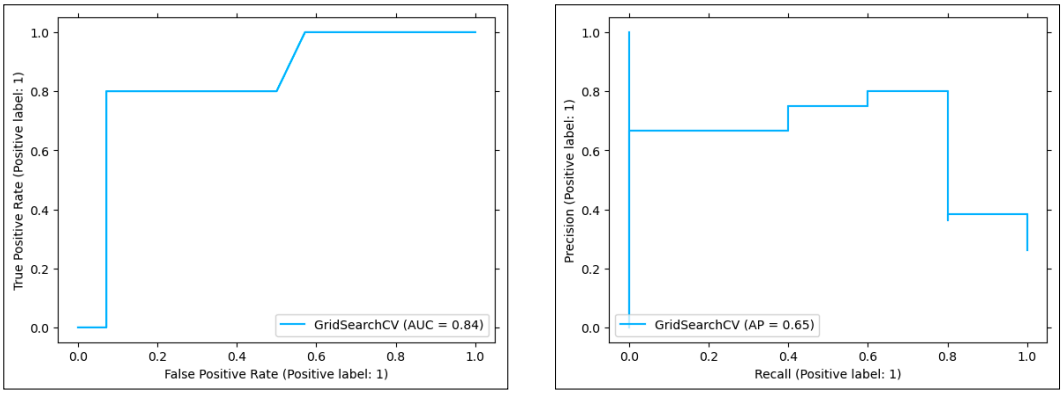


Figure 48. ROC (left) and Precision Recall (right) Curves – Final Best Combined Falls Model

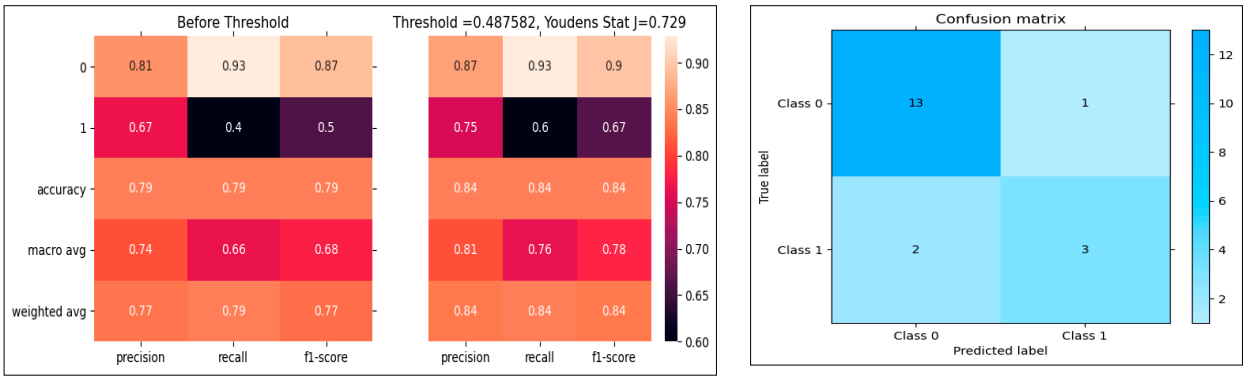


Figure 49. Classification Report (left) and Classification Matrix (right) – Final Best Combined Falls Model

As we can note, the initial best models improved after further applying feature selection to check impact of number of features (Table 8).

Table 8. Impact of number of features on model performance. (Non-stairs data - 20% split)

Stairs	Data	Model	Initial Best Model Performance		Final Best Model Performance	
			No. of features	ROC-AUC	No. of features	ROC-AUC
Ascent	GANN	XGB	30	0.90	27	0.90
Descent	GANN	LRCV	30	0.85	67	0.94
Combined	GANN	XGB	30	0.80	19	0.84

While Ascent Fall performance remained same, Descent Fall model improved significantly but the trade-off here was number of features were increased as well, but is not a major issue considering the size of the data. Combined fall showed slight improvement but as expected is not ideal to use to predict for both kind of falls.

Although these results showed good performance, it should be noted that, the test data size is extremely small and considering the class imbalance / very few positive samples in test (Table 9), the test data may not be representative of the data the model might encounter in future, thereby not accurately reflecting on the model performance and thereby misleading the outcome. Hence an alternative size test set (30% split) was necessary to explore if the output performance get significantly affected.

Table 9. Test Data Fall Counts (Non Stairs Data)

Stairs	20% Test Split		30% Test Split	
	Fall	No-Fall	Fall	No-Fall
Ascent	4	15	6	23
Descent	2	17	3	26
Combined	5	14	8	21

5.4.2 NON STAIRS DATA – 30% SPLIT

Similar to the previous 20% split set, 30% split set also followed similar experiment process with 2 phases – all features and top 30, comparing the models and checking best models for further feature impact. Additionally, to check if size of training data plays a role in model output, additional 2 GANN based samples were also used in this case. The outcome resulted in reduced model scores for all fall categories in both the phases with no ROC-AUC > 0.80 irrespective of the falls and oversampling strategies.

Table 10. Best Model Comparisons - All vs Top 30 Features (Non Stairs Data - 30% split)

	All Features				Top 30 Features			
Stairs	**Data**	**Model**	**ROC-AUC**	**AP**	**Data**	**Model**	**ROC-AUC**	**AP**
Ascent	GANN1000	RF	0.80	0.69	original	LRCV	0.80	0.68
Descent	Smote	RF	0.75	0.60	smote	LRCV	0.71	0.49
Combined	Random	RF	0.76	0.65	smote	LRCV	0.75	0.65

A pattern with best models was observed in terms of algorithms used, where when all features were used, Random Forrest with different sampled data produced the best results, whereas when logistic regression with cross validation worked best with 30 best features.

XGBoost is prone to overfitting when too many trees are used in the model or if it is trained on a smaller dataset, which can likely be a reason XGBoost didn't perform well for 30% split. Also, when making predictions, XGBoost is largely unable to extrapolate target values beyond the limits of training data which is likely scenario given the training set has very few samples.

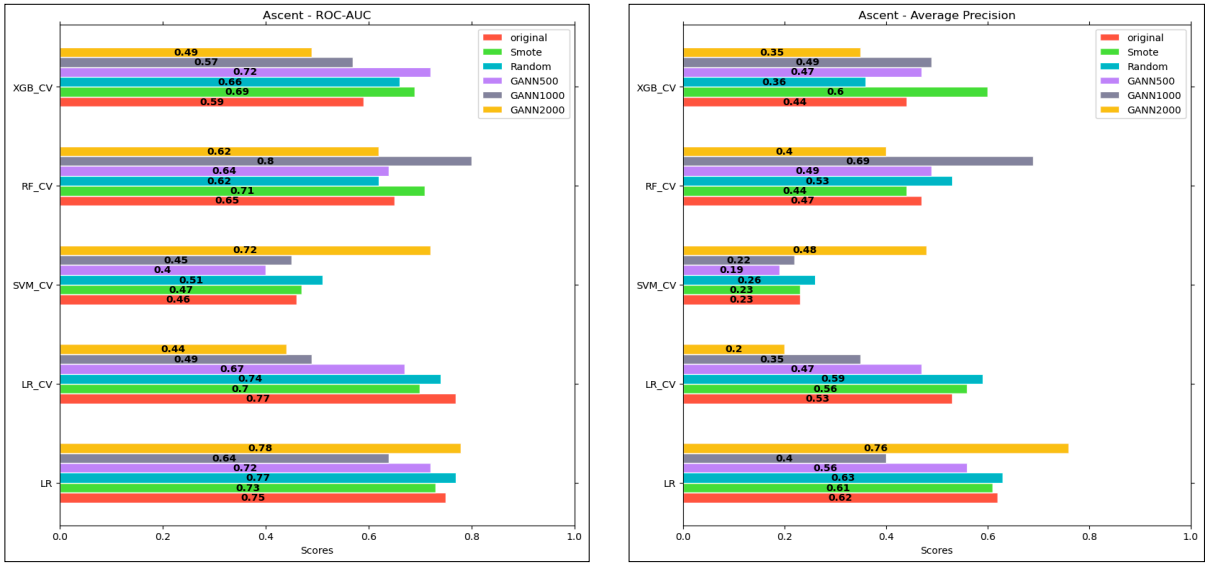


Figure 50. Model Scores - Ascent Falls - Non Stairs Data (30% split, All Features)

Additionally, apart from for Ascent Falls, GANN overall didn't perform which likely may be due to less training data for synthesizing and further less relatability with sampled data and test data.

Best model performances were observed when all features were used. These best models were further tried for impact of number of features (Figure 51) similar to the previous 20% split, and the scores improved (Table 11). No stable trend was observed with the scores while varying features (too many peaks / fluctuations) especially with Ascent, unlike what was observed with 20% split set.

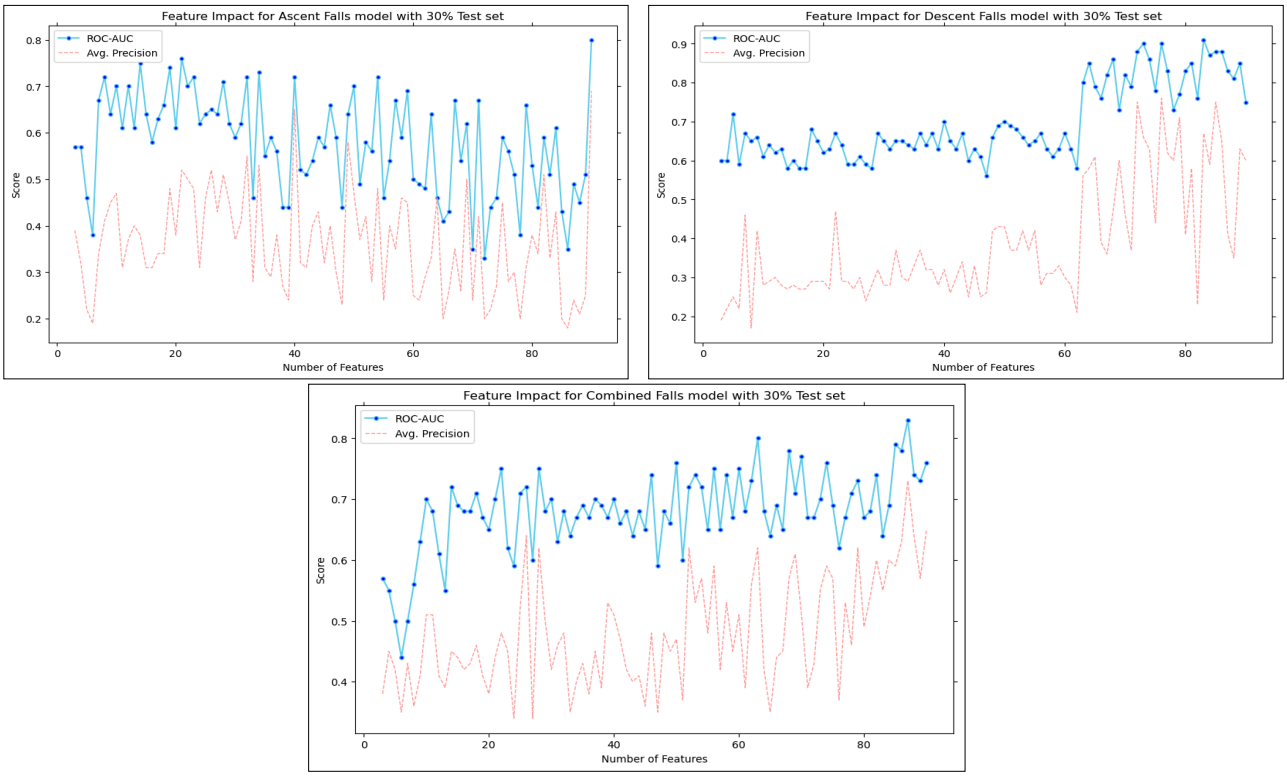


Figure 51. Feature Impact on best models for respective fall category (Non Stairs Data - 30% split)

Table 11. Impact of number of features on model performance. (Non-stairs data - 30% split)

Stairs	Data	Model	Initial Best Model Performance		Final Best Model Performance	
			No. of features	ROC-AUC	No. of features	ROC-AUC
Ascent	GANN1000	RF	All	0.80	All	0.80
Descent	Smote	RF	All	0.75	83	0.91
Combined	Random	RF	All	0.76	87	0.83

Even though model for descent falls improved tremendously with AUC from 0.75 to 0.91, the rest did not much especially, model for Ascent falls showed same performance irrespective of the features.

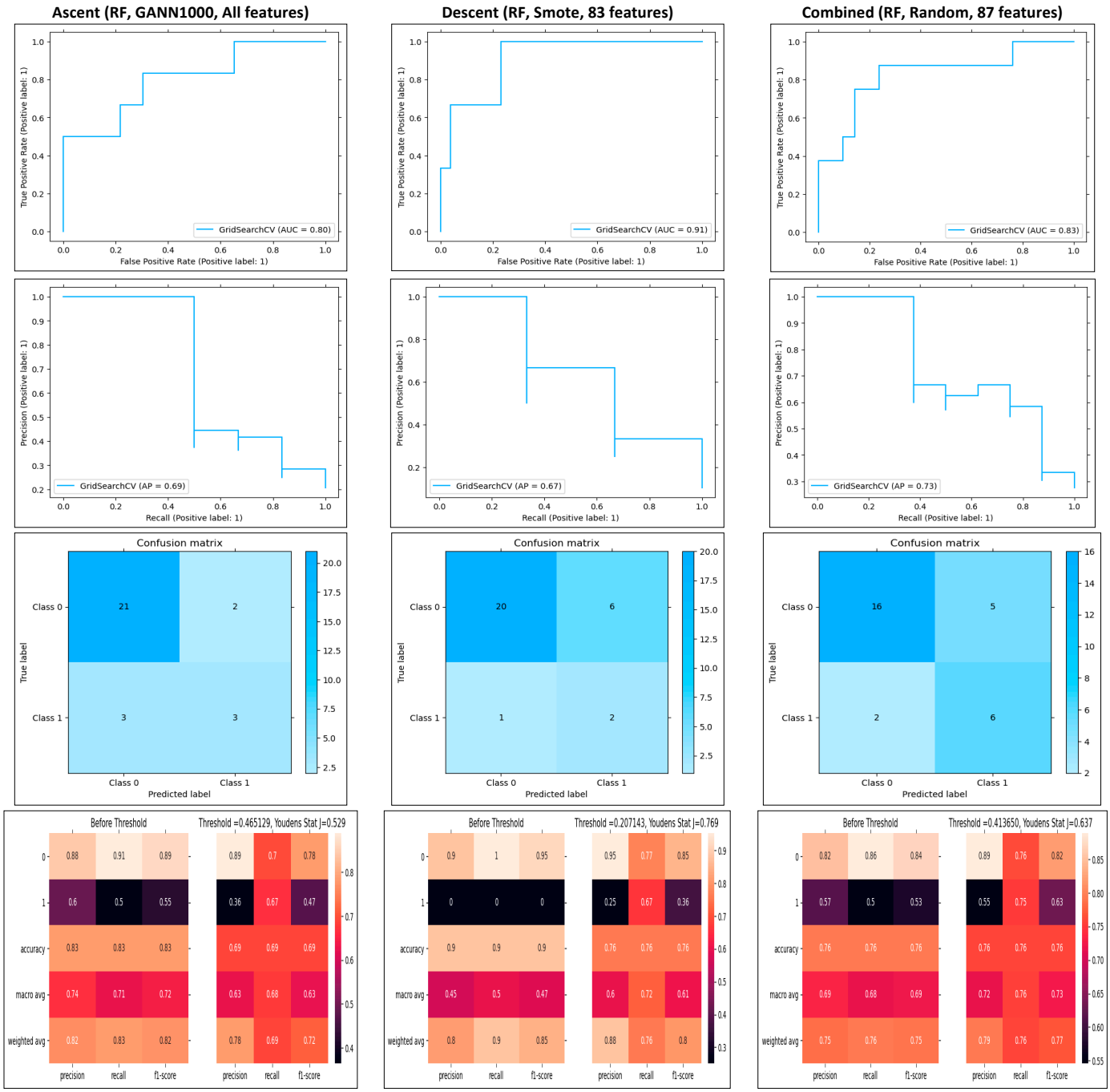


Figure 52. Final Best Model Score - Plots (Non Stairs Data, 30% Split)

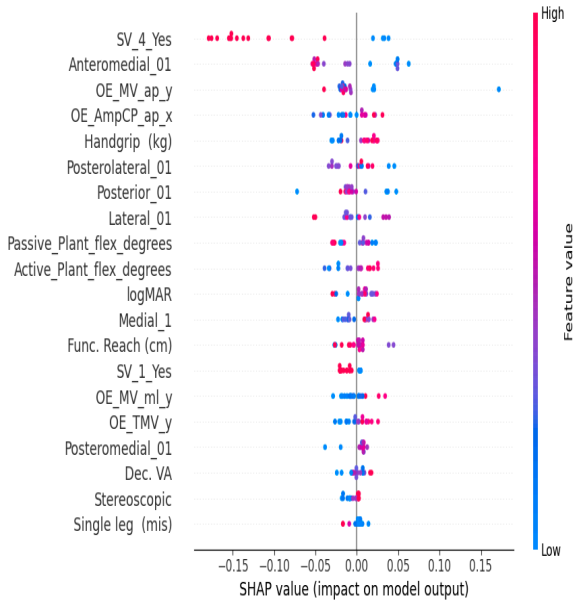
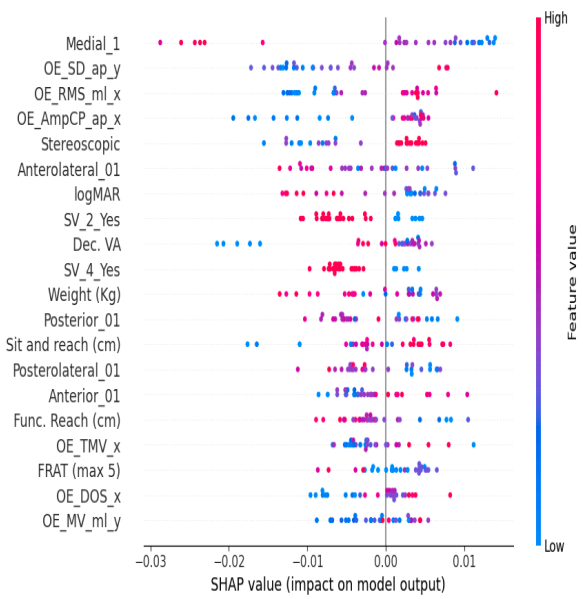
5.4.3 COMPARING THE FINAL BEST MODELS – 20% VS 30%

Table 12. Final Best Models - Comparison & Findings(Non Stairs Data)

	20 % Test Split				30 % Test Split			
Stair Category	**Data Sample**	**Model**	**Features**	**ROC-AUC**	**Data Sample**	**Model**	**Features**	**ROC-AUC**
Ascent	GANN2000	XGB	27	0.90	GANN1000	RF	All	0.80
Descent	GANN2000	LRCV	67	0.94	Smote	RF	83	0.91
Combined	GANN2000	XGB	19	0.84	Random	RF	87	0.83
Key Takeaway	1) Oversampling was necessary for best performing models for each of the fall categories.							
	2) With 30% Test splits, the best models needed more than 90% of total features as input whereas, with 20% split, apart from Descent fall prediction, less than 30% of total features were required to get the best model score.							
	3) Additionally, descent fall models required more than 70% features in both cases, which might likely be due the fact that descent fall positive classes were very few. Also, in both cases although score was above 0.90, this can still be misleading given there are only maximum 3 positive samples for descent fall.							
	4) GANN based sampling contributed to all best models in 20% split set. This can likely be due to more availability of training data for synthetic data generation. This in turn allows more training samples which can relate to similar properties of the test samples allowing model to learn and predict better.							
	5) Comparatively with 30% split, only GANN1000 with random forest worked average with Ascent and this may be due to more positive ascent fall samples available in train set compared to descent. However, the could not improve by altering the number of features. This also calls for change in sampling strategy and then checking feature impact for this case.							
	6) Comparing all the models executed, SVM did not fare well for with any sample or fall category which may likely be due to the soft margin optimization problem i.e. bias towards majority class or there is more noise within data i.e. target classes are overlapping. Similarly, logistic regression without cross validation was not involved in best results which was expected.							
	7) Logistic Regression with cross validation worked well with descent indication linear separability or linear relationships. Additionally the 2 decision tree based algorithms performed well as expected as can detect linear as well as non-linear patters within data.							
	8) Predicting ascent and descent by combining them together is not ideal with give data.							
	9) The scores attained for predicting each fall was partly sufficient enough to suggest Non Stairs Data can be useful for stair fall prediction and the claim can be justified if additional true data samples are made available.							

5.4.4 SHAP ANALYSIS – NON STAIRS DATA

SHAP summary plots for best models for each split is compared to find if the features contributing in 20% split set data remains same if the train size decreases, for each fall category.

Best Model SHAP Plot- 20% Test Split	Best Model SHAP Plot- 30% Test Split
	

Data Collections	Features	Data Collections	Features
Vision Test	SV_4_Yes, SV_1_Yes, Sterioscopic, logMAR, Dec. VA	Vision Test	Stereoscopic, logMAR, SV_2_Yes, Dec. VA, SV_4_Yes
Star Excursion Balance Test	Anteromedial_01, Posterolateral_01, Posterior_01, Lateral_01, Medial_1, Posteromedial_01	Star Excursion Balance Test	Medial_1, Anterolateral_01, Posterior_01, Posterolateral_01, Anterior_01
Postural Balance Data (Statistical Measures)	OE_MV_ap_y, OE_AmpCP_ap_x, OE_MV_ml_y, OE_TMV_y	Postural Balance Data (Statistical Measures)	OE_SD_ap_y, OE_RMS_ml_x, OE_AmpCP_ap_x, OE_TMV_x, OE_DOS_x, OE_MV_ml_y
Functional Test	Handgrip (kg), Func. Reach (cm), Single leg (mis)	Functional Test	Sit and reach (cm), Func. Reach (cm)
Range of Motion Test	Passive_Plant_flex_degrees, Active_Plant_flex_degrees	Questionnaires	FRAT (max 5)
		General Info	Weight(KG)

Figure 53. SHAP Summary Plot - Ascent Fall Model (Non Stairs Data)

From above figure (Figure 53) for Ascent Fall Models, we can note that the features contributing to one model output doesn't have same impact or have no impact on the other model output. SV_4_yes, a vision test encoded parameter, has significant negative impact (resulting in no fall prediction) on the 20% split model output when is has a high value (1 in this case), where as it does not contribute with same impact with 30% split. There are features like Weight which is impacting the model with 30% split however has no significant contribution with 20% split. Note that both the

models are different so it is expected to produce different contributions by each feature. Some features are common in both which contributes significantly.

One key pattern observed was that several features from Vision Test, Star Excursion Balance Test, Postural balance Measures, Functional Test and Range of Motion Test have significantly contributed to the model output for all the fall categories irrespective of the split size.

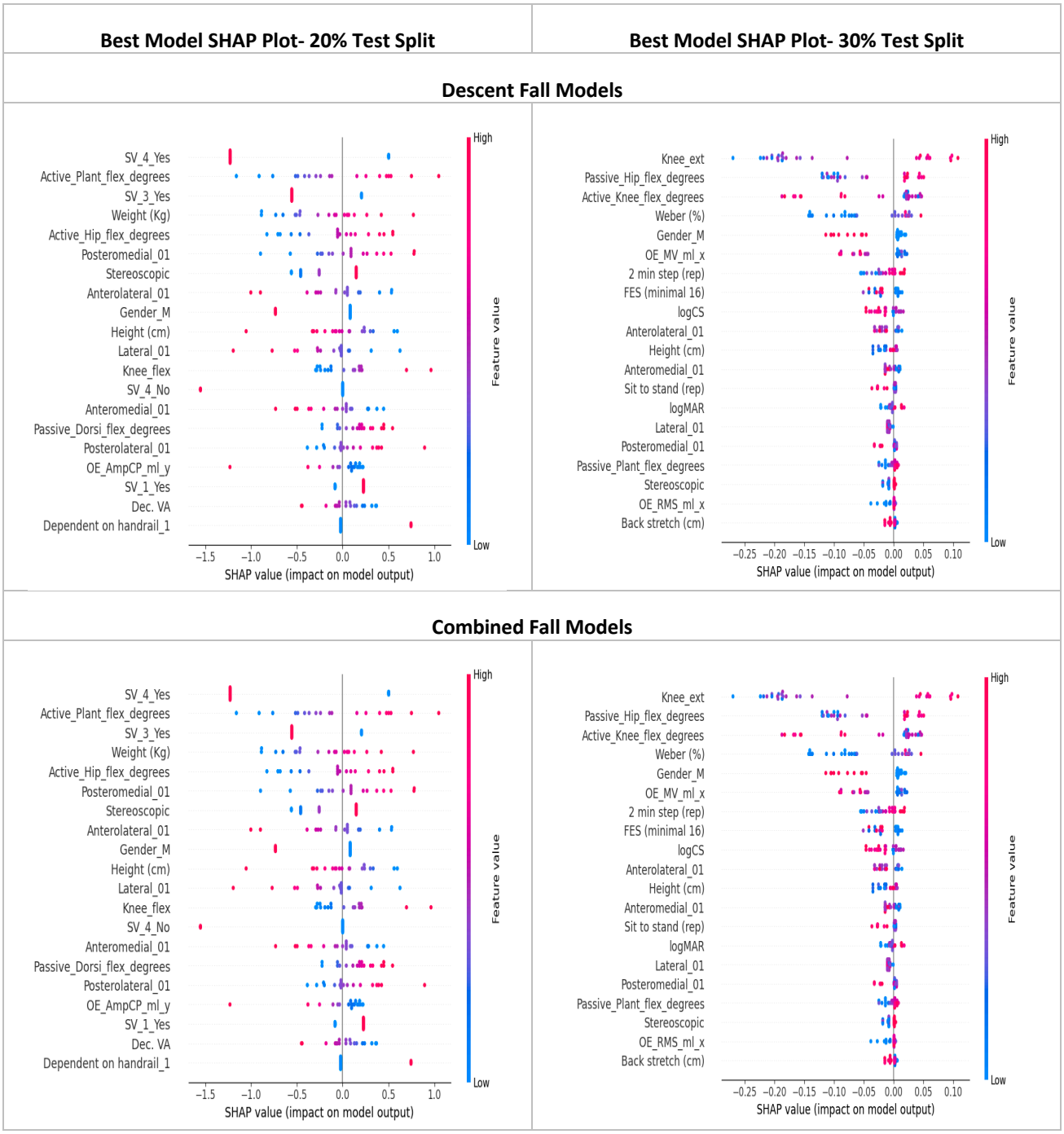


Figure 54. SHAP Summary Plots - Descent and Combined Fall Models (Non Stairs Data)

5.4.5 STAIRS + NON STAIRS DATA – 30% SPLIT

Unlike Non Stairs data, this data set was used with only logistic regression with cross validation and results as follows:

Table 13. Best Model Scores & key findings - Stairs + Non Stairs Data

Stairs	Data Sample	Model	No. of Features	ROC-AUC	AP
Ascent	Smote	LR_CV	4	0.85	0.72
Descent	Random	LR_CV	38	0.97	0.83
Combined	random	LR_CV	29	0.78	0.72
Key Takeaway	1) Ascent Fall model gave best score with 4 best non-stairs related features while descent Fall model produces the best results with use of stairs data. But once again this can be misleading as the test set contains only 2 positive cases while 19 negative cases.				
	2) Combining the two falls was once again proved to be unsuccessful.				
	3) Oversampling was necessary to get best results				
	4) Stair based data contributed to the descent fall model output especially the foot clearance variance, affecting negatively with higher values.				

Ascent **Descent** **Combined**

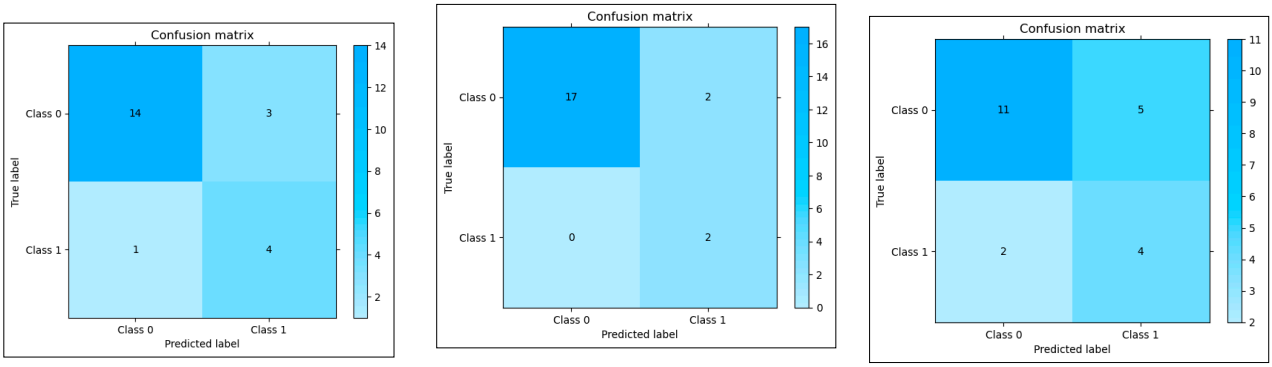


Figure 55. Classification Matrix for Best Models – (Stairs + Non Stairs Data)

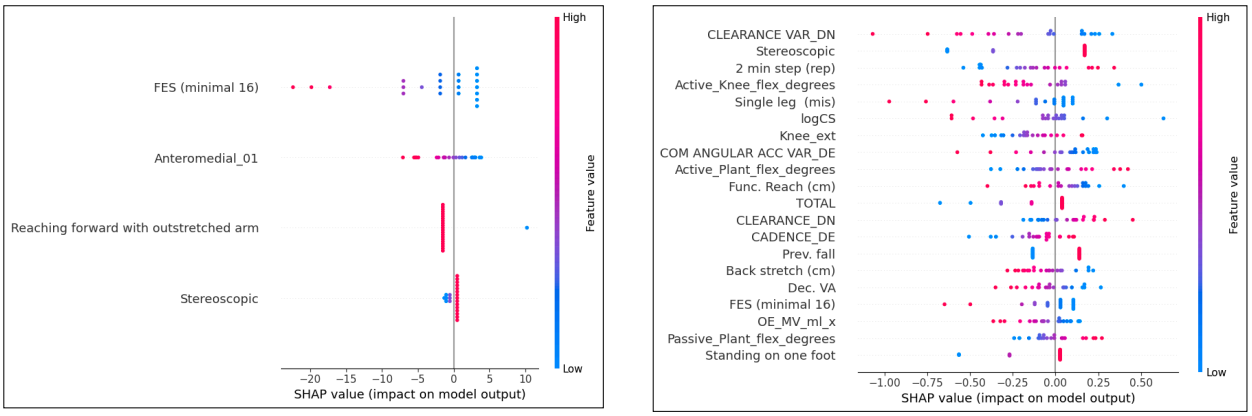


Figure 56. SHAP Summary Plots (Ascent - Descent) (Stairs + Non Stairs Data)

6 CONCLUSION

Stair fall prediction is somewhat an uncharted territory open for exploration. Machine learning techniques have been employed in various studies however, mainly limited to unsupervised approaches. This supervised ML based study was aimed to provide an alternative which can be used predict stair falls thereby preventing related problems. The study began with a primary focus on utilizing non – stairs related data for the purpose, however the available data limitations in terms of size and class imbalance posed a major problem but was partially tackled using synthetic data based oversampling techniques. Oversampling the data to balance the classes was discovered to be an absolute necessity for better predictions. GANN based sampling technique was found effective when 80% training data was present for synthetic data generation.

Another obstacle was efficiently executing multiple models, which would be tuned differently each time for the multiple data inputs, based on multiple sampling, for the three fall categories, in addition to application of feature selection to check how number of input features at a time could affect the prediction. The approach of trying the model first with all features followed by one third of features, and then, further testing the initial best model among the two having optimal parameters, with 3 to 'n' feature inputs was effective, resulting in improved model performance without being computationally expensive. However, there is still a question here as in why limit the initial modelling to just - all or 1/3 features, which can be a further work.

Non stairs data as a standalone was discovered to be useful for predicting ascent and descent stair falls, with several features from Vision Test, Star Excursion Balance Test, Postural balance Measures, Functional Test and Range of Motion Test, having significant contribution towards the prediction. This can eliminate the use of biomechanical features (stair data) entirely, avoiding the Laboratory based expenses to maintain a biomechanical staircase. However, given the size of test set, this needs to be further improved and justified if more real world data is made available. Similarly, Stairs + Non Stairs data also provided efficient performance, especially for descent falls, given only one model was used. This can similarly be tried for more real data and more models in future. Applying other available oversampling techniques or proficiently using CTGAN for conditional data generation or improving existing synthetic data quality can also be explored ahead with this study.

If sufficient real world data is made available, an in-depth exploration to efficiently predict stair falls, is bound to bring a breakthrough that can affect lives of people.

7 SELF – EVALUATION

The project so far was challenging because of the dataset. More domain knowledge was needed which could have brought forward alternate strategies to tackle the problem. The project exposed me to new techniques like GAN and various python libraries for coding which could be beneficial for my future career.

However some major challenger were faced during the project as follows:

a) Domain Specific : No data dictionary was made available which posed a major issue during early stages.
b) Computational Specific : The computing work was conducted on personal system hence running the pipeline, especially hyperparameter tuning required plenty of time to execute. Additionally, GANN techniques were more computationally heavy and needed cloud based services like Kaggle and Google Collab which, when used, limited use of GPU time.
c) Coding Challenges : Creating a generic code which could be used in a variety of ways was a hassle and time consuming.
d) Human Error : Error while coding posed serious issues so much so that once a problem took more than 3 weeks to get resolved.

To tackle the initial project timeline was adjusted accordingly based on necessities. Consulting supervisors when in need helped extensively. Planning what to code based on needs was much needed help. Some challenges could not be avoided and had to face it head on and learn from mistakes, which in turn enabled me to strengthen mentally to manage academic stress efficiently.

8 REFERENCES

[1] N.I.C.E, “Falls - Risk Assessment,” 2019. [Online]. Available: https://cks.nice.org.uk/topics/falls-risk-assessment/. [Accessed 30 August 2023].

[2] NHS, “Falls,” 2021. [Online]. Available: https://www.nhs.uk/conditions/falls/. [Accessed 30 August 2023].

[3] Startzell JK, Owens DA, Mulfinger LM, Cavanagh PR. Stair negotiation in older people: a review. J Am Geriatr Soc. 2000;48(5):567-580. doi:10.1111/j.1532-5415.2000.tb05006.x

[4] Physiopedia, “Falls Risk Assessment Tool (FRAT),” [Online]. Available: https://www.physio-pedia.com/Falls_Risk_Assessment_Tool_(FRAT). [Accessed 30 August 2023].

[5] Ackermans TMA, Francksen NC, Casana-Eslava RV, et al. A novel multivariate approach for biomechanical profiling of stair negotiation. Exp Gerontol. 2019;124:110646. doi:10.1016/j.exger.2019.110646

[6] Ackermans T, Francksen N, Lees C, et al. Prediction of Balance Perturbations and Falls on Stairs in Older People Using a Biomechanical Profiling Approach: A 12-Month Longitudinal Study. J Gerontol A Biol Sci Med Sci. 2021;76(4):638-646. doi:10.1093/gerona/glaa130

[7] Clark TG, Bradburn MJ, Love SB, Altman DG. Survival analysis part I: basic concepts and first analyses. Br J Cancer. 2003;89(2):232-238. doi:10.1038/sj.bjc.6601118

[8] Wikipedia, “K-means clustering,” [Online]. Available: https://en.wikipedia.org/wiki/K-means_clustering. [Accessed 30 August 2023].

[9] Machine Learning Mastery, “A Gentle Introduction to Generative Adversarial Networks (GANs),” 2019. [Online]. Available: https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/. [Accessed 30 August 2023].

[10] Branco, Paula & Torgo, Luís & Ribeiro, Rita. A Survey of Predictive Modelling under Imbalanced Distributions. 2015.

[11] T. Hasanin and T. Khoshgoftaar, “The Effects of Random Undersampling with Simulated Class Imbalance for Big Data,” 2018 IEEE International Conference on Information Reuse and Integration (IRI), Salt Lake City, UT, USA, 2018, pp. 70-79, doi: 10.1109/IRI.2018.00018.

[12] S. K. M, Devidas, S. N. Pai, S. Kolekar, V. Pai and B. R, "Use of Machine Learning and Random OverSampling in Stroke Prediction," 2022 International Conference on Artificial Intelligence and Data Engineering (AIDE), Karkala, India, 2022, pp. 331-337, doi: 10.1109/AIDE57180.2022.10060313.

[13] Huang, Pengtao. Classification of Imbalanced Data Using Synthetic Over-Sampling Techniques. 2015. URL: https://api.semanticscholar.org/CorpusID:55362853.

[14] L. K. Xin and N. b. A. Rashid, "Prediction of Depression among Women Using Random Oversampling and Random Forest," 2021 International Conference of Women in Data Science at Taif University (WiDSTaif), Taif, Saudi Arabia, 2021, pp. 1-5, doi: 10.1109/WiDSTaif52235.2021.9430215.

[15] S. Harjai, S. K. Khatri and G. Singh, "Detecting Fraudulent Insurance Claims Using Random Forests and Synthetic Minority Oversampling Technique," 2019 4th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, 2019, pp. 123-128, doi: 10.1109/ISCON47742.2019.9036162.

[16] Zhenhao Jiang, Tingting Pan, Chao Zhang, Jie Yang. A New Oversampling Method Based on the Classification Contribution Degree. Symmetry. 2021;13(2):194. doi:10.3390/sym13020194

[17] Google Machine Learning, “Overview Of GAN Structure,” [Online]. Available: https://developers.google.com/machine-learning/gan/gan_structure. [Accessed 30 August 2023].

[18] Wikipedia, “Generative Adversarial Networks,” [Online]. Available: https://en.wikipedia.org/wiki/Generative_adversarial_network. [Accessed 30 August 2023].

[19] V. s. Krishna Katta, H. Kapalavai and S. Mondal, "Generating New Human Faces and Improving the Quality of Images Using Generative Adversarial Networks(GAN)," 2023 2nd International Conference on Edge Computing and Applications (ICECAA), Namakkal, India, 2023, pp. 1647-1652, doi: 10.1109/ICECAA58104.2023.10212099.

[20] Sauber-Cole, R., Khoshgoftaar, T.M. The use of generative adversarial networks to alleviate class imbalance in tabular data: a survey. J Big Data 9, 98 (2022). https://doi.org/10.1186/s40537-022-00648-6

[21] Sharma, Anuraganand & Singh, Prabhat & Chandra, Rohitash. (2022). SMOTified-GAN for Class Imbalanced Pattern Classification Problems. IEEE Access. 10. 1-1. 10.1109/ACCESS.2022.3158977.

[22] Cheon, Francis & Dong, Hee & Lee, & Park, Ji & Choi, Hye & Lee, Jun & Lee, Ook. (2021). Ctgan Vs Tgan? Which One Is More Suitable For Generating Synthetic Eeg Data. 99.

[23] H. Tan et al., "Tabular GAN-Based Oversampling of Imbalanced Time-to-Event Data for Survival Prediction," 2023 8th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA), Chengdu, China, 2023, pp. 376-380, doi: 10.1109/ICCCBDA56900.2023.10154883.

[24] Synthetic Data Vault, “CTGANSynthesizer,”. [Online]. Available: https://docs.sdv.dev/sdv/single-table-data/modeling/synthesizers/ctgansynthesizer. [Accessed 30 August 2023].

[25] Bolón-Canedo, V., Sánchez-Maroño, N. & Alonso-Betanzos, A. A review of feature selection methods on synthetic data. Knowl Inf Syst 34, 483–519 (2013). https://doi.org/10.1007/s10115-012-0487-8

[26] Machine Learning Mastery, “How to Perform Feature Selection With Numerical Input Data,” 2020. [Online]. Available: https://machinelearningmastery.com/feature-selection-with-numerical-input-data/. [Accessed 30 August 2023].

[27] G. Zhu, H. Zhao, H. Liu and H. Sun, "A Novel LSTM-GAN Algorithm for Time Series Anomaly Detection," 2019 Prognostics and System Health Management Conference (PHM-Qingdao), Qingdao, China, 2019, pp. 1-6, doi: 10.1109/PHM-Qingdao46334.2019.8942842.

[28] Bach M. The Freiburg Visual Acuity Test – automatic measurement of visual acuity. Optometry Vision Sci. 1996;73(1):49–53.

[29] Andrews AW, Thomas MW, Bohannon RW. Normative values for iso- metric muscle force measurements obtained with hand-held dynamometers. Phys Ther. 1996;76:248–259. doi:10.1093/ptj/76.3.248

[30] Norkin CC, White DJ. Measurement of Joint Motion: A Guide to Goniometry. FA Davis Company; 2016.

[31] Rikli RE, Jones CJ. Development and validation of a functional fitness test for community-residing older adults. J Aging Phys Activ. 1999;7(2):129– 161. doi:10.1123/japa.7.2.129

[32] Berg KO, Wood-Dauphinee SL, Williams JI, Maki B. Measuring balance in the elderly: validation of an instrument. Can J Public Health. 1992;83(suppl 2):S7–S11.

[33] Physiopedia, “Star Excursion Balance Test,” [Online]. Available: https://www.physio-pedia.com/Star_Excursion_Balance_Test. [Accessed 30 August 2023].

[34] Analytics Vidya, Mayur Badole, “Difference Between fit(), transform(), and fit_transform() Methods in Scikit-Learn,” [Online]. Available: https://www.analyticsvidhya.com/blog/2021/04/difference-between-fit-transform-fit_transform-methods-in-scikit-learn-with-python-code/. [Accessed 30 August 2023].

[35] Synthetic Data Vault, “Evaluation,”. [Online]. Available: https://docs.sdv.dev/sdv/single-table-data/evaluation. [Accessed 30 August 2023].

[36] Synthetic Data Vault, “SD Metrics - What’s Included?,”. [Online]. Available: https://docs.sdv.dev/sdmetrics/reports/quality-report/whats-included. [Accessed 30 August 2023].

[37] Synthetic Data Vault, “SD Metrics – TVComplement,”. [Online]. Available: https://docs.sdv.dev/sdmetrics/metrics/metrics-glossary/tvcomplement. [Accessed 30 August 2023].

[38] Synthetic Data Vault, “SD Metrics – CorrelationSimilarity,”. [Online]. Available: https://docs.sdv.dev/sdmetrics/metrics/metrics-glossary/correlationsimilarity. [Accessed 30 August 2023].

[39] Synthetic Data Vault, “SD Metrics – ContigencySimilarity,”. [Online]. Available: https://docs.sdv.dev/sdmetrics/metrics/metrics-glossary/contingencysimilarity. [Accessed 30 August 2023].

[40] Pypi, “Table Evaluator,”. [Online]. Available: https://pypi.org/project/table-evaluator/. [Accessed 30 August 2023].

[41] Towards Data Science. Miriam Santos, “How to Generate Real-World Synthetic Data using CTGAN,”. [Online]. Available:https://towardsdatascience.com/how-to-generate-real-world-synthetic-data-with-ctgan-af41b4d60fde. [Accessed 30 August 2023].

[42] SHAP, “SHAP Documentation,”. [Online]. Available: https://shap.readthedocs.io/en/latest/. [Accessed 30 August 2023].

9 APPENDIX

9.1 CODE SNIPPETS

A. TRAIN TEST SPLIT

The train data and test data split was then performed using 'train_test_split' method from scikit learn python library.

```
def split(self, X,y, test_size = 0.0):

    if test_size == 0:
        test_size = self.test_split_size

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = test_size, random_state=42, stratify=y)

    return X_train, X_test, y_train, y_test
```

B. IMPUTATION

The instances with missing values were than imputed accordingly using 'SimpleImputer' from scikit learn library.

```
def fill_na(self, train, test, strategy='mean'):

    imp = SimpleImputer(missing_values=np.nan, strategy=strategy)
    imp.fit(train.values.reshape(-1,1))

    return imp.transform(train.values.reshape(-1,1))[:,0], imp.transform(test.values.reshape(-1,1))[:,0]
```

C. CTGAN DATA SYNTHESIZER

```
from sdv.single_table import CTGANSynthesizer
from sdv.evaluation.single_table import evaluate_quality
from sdv.evaluation.single_table import run_diagnostic

synthesizer = CTGANSynthesizer(metadata,
    embedding_dim = 16,
    generator_dim = (16,16),
    generator_lr = 1e-4,
    discriminator_dim = (16,16),
    batch_size = 30,
    cuda = False,
    verbose=True,
    epochs = 600
    )
synthesizer.get_parameters()
synthesizer.fit(df)

```

D. ONE HOT ENCODING

Encoding done using 'get_dummies' from Pandas Library

```
def getDummies(self, X_train,X_test):

    X_train = pd.get_dummies(X_train,drop_first=True)
    X_test = pd.get_dummies(X_test,drop_first=True)

    print(" X Train-Test Shape after encoding: ",X_train.shape, X_test.shape)

    # Get missing columns in test set from the training test
    missing_cols = set( X_train.columns ) - set( X_test.columns )
    print(missing_cols)
    # Add a missing column in test set with default value equal to 0
    for c in missing_cols:
        X_test[c] = 0
    # Ensure the order of column in the test set is in the same order than in train set
    X_test = X_test[X_train.columns]

    print(" X Train-Test Shape after adding missing columns: ",X_train.shape, X_test.shape)

    return X_train, X_test
```

E. FEATURE IMPORTANCE AND SELECTION

Feature importance score was determined using a feature selection method – 'SelectKBest' from scikit learn library.

```
def select_features(X_train, y_train, X_test,k=30):
    # configure to select all features
    fs = SelectKBest(score_func=f_classif, k=k)
    # learn relationship from training data
    fs.fit(X_train, y_train)

    cols_idxs = fs.get_support(indices=True)
    X_train_fs = X_train.iloc[:,cols_idxs]
    X_test_fs = X_test[list(X_train.columns)]

    return X_train_fs, X_test_fs, fs
```

F. OVERSAMPLING

SMOTE technique can be implemented using 'SMOTE' method from imbalanced-learn python library. Random Oversampling can be implemented using 'RandomOverSampler' method from imbalanced-learn python library.

In random oversampling, examples from the minority class can be chosen and added to the new "more balanced" training dataset multiple times; they are selected from the original training dataset, added to the new training dataset, and then returned or "replaced" in the original dataset, allowing them to be selected again.

In SMOTE A random example/data point from minority class is first selected and 'k' (default=5) nearest neighbours for that example are found. A randomly selected neighbour is chosen and a synthetic example is created at a randomly selected point between the two examples in feature space.

```
def smote_oversample(self, X,y, kn = 5):
    print('==========SMOTE Over Sampling=============')
    # define oversampling strategy
    oversample = SMOTE(sampling_strategy = 'minority', random_state=42, k_neighbors=kn)
    # fit and apply the transform
    X_over, y_over = oversample.fit_resample(X, y)
    # summarize class distribution
    print(Counter(y_over))
    return X_over,y_over

def random_oversample(self, X,y):
    print('==========RANDOM Over Sampling=============')
    # define oversampling strategy
    oversample = RandomOverSampler(sampling_strategy='minority', random_state = 42)
    # fit and apply the transform
    X_over, y_over = oversample.fit_resample(X, y)
    # summarize class distribution
    print(Counter(y_over))
    return X_over,y_over
```

Reference:

Machine Learning Mastery, "Random Oversampling and Undersampling for Imbalanced Classification," 2020. [Online]. Available: https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/. [Accessed 30 August 2023].

Machine Learning Mastery, "SMOTE for Imbalanced Classification for Python," 2021. [Online]. Available: https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/. [Accessed 30 August 2023].

G. FEATURE SCALING

'StandardScaler' method from scikit-learn library was used to normalise the data.

```
scaler = StandardScaler()
X_train = pd.DataFrame(scaler.fit_transform(X_train), index=X_train.index, columns=X_train.columns)
X_test = pd.DataFrame(scaler.transform(X_test), index=X_test.index, columns=X_test.columns)
```

H. DIMENSIONALITY REDUCTION

```
# Perform dimensionality reduction on the dataset
# perform PCA, UMAP ans tSNE for visualisation purpose
def ml_visualisation(X_train,y_train,perplexity=70,title=""):
    # PCA
    pca = PCA(n_components=2)
    X_pca = pca.fit_transform(X_train)

    #TSNE
    tsne = TSNE(n_components=2, perplexity=perplexity, random_state=42)
    X_tsne = tsne.fit_transform(X_train)

    #UMAP
    umap_model = umap.UMAP(n_components=2, random_state=42, n_neighbors = 3)
    X_umap = umap_model.fit_transform(X_train)

    # Plot the visualization
    plt.figure(figsize = (25, 10))
    plot_data_projection(X_pca, y_train, 3,1,"PCA")
    plot_data_projection(X_tsne, y_train, 3,2,"tSNE")
    plot_data_projection(X_umap, y_train, 3,3,"UMAP")
    plt.suptitle(title,size=14)
    return plt
```

I. LOGISTIC REGRESSION

```
def model_A(X_train, X_test, y_train, y_test, text):

    print("="*20+text+"="*20)

    # Logistic Regression with Default Parameters
    mdl = LogisticRegression(max_iter = 100)    # sets up the algorithm (or learner)
    mdl.fit(X_train, y_train)

    # Predicts output and reports the model scores
    plot_report(mdl, X_test, y_test,text)
```

J. LOGISTIC REGRESSION WITH CROSS VALIDATION

```
def model_A(X_train, X_test, y_train, y_test, text):

    print("="*20+text+"="*20)

    mdl = LogisticRegressionCV(penalty='l2',
                               scoring='roc_auc',
                               cv=5,
                               Cs = 10) #grid of Cs values are chosen in a logarithmic scale between 1e-4 and 1e4.
    mdl.fit(X_train, y_train)


    plot_report(mdl, X_test, y_test,text)
```

K. SUPPORT VECTOR CLASSIFIER

The parameters were tested for range a of values to find the best fit, avoid overfitting and improve accuracy. Major parameter include – (i) the penalty parameter of error term 'C' which controls the trade-off between achieving a low training error and a low testing error, (ii) the kernel parameter to specify the kernel functions which can be radial basis function, liner or polynomial, (iii) kernel specific parameters like gamma parameter for rbf to control the distance of the influence of a single training point, degree parameters to specify the degree of polynomial functions.

```
def model_A(X_train, X_test, y_train, y_test, text):

    print("="*20+text+"="*20)

    svm_hyp_grid = [
    {'C': [0.1, 1, 10, 100], 'kernel': ['linear']},
    {'C': [0.1, 1, 10, 100], 'degree': [2, 4, 8], 'kernel': ['poly']},
    {'C': [0.1, 1, 10, 100], 'gamma': [0.01, 0.1, 0.5, 1, 100], 'kernel': ['rbf']}
]

    mdl = GridSearchCV(SVC(probability=True), param_grid=svm_hyp_grid, cv=5, scoring='roc_auc').fit(X_train, y_train)
    mdl.fit(X_train, y_train)

    print("Best Params: ",mdl.best_params_)

    plot_report(mdl, X_test, y_test,text)
```

L. RANDOM FORREST CLASSIFIER

```
def model_A(X_train, X_test, y_train, y_test, text):

    print("="*20+text+"="*20)

    n_estimators = [10,15,20,25,30,35,40]
    max_depth = [3,5,7,9,11]
    min_samples_leaf = range(2,18,2)
    bootstrap = [True, False]

    param_grid = {
        "n_estimators": n_estimators,
        "max_depth": max_depth,
        'max_features':[0.7, 0.9, 1],
        "min_samples_leaf": min_samples_leaf,
        "bootstrap": bootstrap,
    }

    rf = RandomForestClassifier(random_state=42)

    mdl = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, n_jobs=-1, scoring='roc_auc')
    mdl.fit(X_train, y_train)

    print("Using hyperparameters --> \n", mdl.best_params_)
    print("Best Score --> \n", mdl.best_score_)

    plot_report(mdl, X_test, y_test,text)
```

M. XGBOOST CLASSIFIER

```
def model_A(X_train, X_test, y_train, y_test, text):

    print("="*20+text+"="*20)


    param_grid = {
        'min_child_weight': [1, 5, 10],
        'gamma': [0.5, 1, 1.5, 2, 5],
        'subsample': [0.6, 0.8, 1.0],
        'colsample_bytree': [0.6, 0.8, 1.0],
        'max_depth': [2, 3, 4]
    }


    xgb = XGBClassifier(learning_rate=0.01, n_estimators=500, booster='gbtree', objective='binary:logistic')

    mdl = GridSearchCV(estimator=xgb, param_grid=param_grid, cv=5, n_jobs=-1, scoring='roc_auc')

    mdl.fit(X_train, y_train)


    print("Using hyperparameters --> \n", mdl.best_params_)
    print("Best Score --> \n", mdl.best_score_)

    plot_report(mdl, X_test, y_test,text)
```

N. MODEL EVALUATION METRICS

```
target_names = ["Non-Fall","Fall"]

y_pred = mdl.predict(X_test)
y_pred_prob = mdl.predict_proba(X_test)[:,1]

# ROC AUC
print("ROC-AUC: ",round(roc_auc_score(y_test, y_pred_prob),2))
# Average Precision/PR AUC
print("Average Precission: ",round(average_precision_score(y_test,
    y_pred_prob),2))
RocCurveDisplay.from_estimator(mdl, X_test, y_test)
# PR Curve
PrecisionRecallDisplay.from_estimator(mdl, X_test, y_test)
plt.show()
```

```
# Classification Report
fig = plt.figure(figsize=(10,4))
plt.subplot(1, 2, 1)
clf_report = classification_report(y_test, y_pred, output_dict=True)
sns.heatmap(pd.DataFrame(clf_report).iloc[-4:-1, :].T, annot=True, cbar
    =False)
plt.yticks(rotation=0)
plt.title('Before Threshold')

fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)

# Use Youden's J statistic to find the optimal threshold
J = tpr - fpr
ix = np.argmax(J)

# Use the above cutoff to predict the class
y_class_opt = np.where(y_pred_prob > thresholds[ix], 1, 0)

plt.subplot(1, 2, 2)
clf_report = classification_report(y_test, y_class_opt, output_dict
    =True)
sns.heatmap(pd.DataFrame(clf_report).iloc[-4:-1, :].T, annot=True,
    yticklabels=False)
plt.yticks()
```

```
# before and after threshold
a = plot.ClassificationReport.from_raw_data(y_test, y_pred,
    target_names=target_names)
plt.close()
b = plot.ClassificationReport.from_raw_data(y_test, y_class_opt,
    target_names=target_names)
plt.close()
a + b

# Confusion matrix
#  before threshold
cr1 = plot.ConfusionMatrix.from_raw_data(y_test, y_pred)

# after threshold
cr2 = plot.ConfusionMatrix.from_raw_data(y_test, y_class_opt)

# Compare
cr1 + cr2
```

O. SHAP PLOTS

Tree Explainer for Random Forrest Model

```
explainer = shap.explainers.Tree(combined_model, combined_X_train,feature_perturbation="interventional",model_output="probability")
shap_values = explainer.shap_values(combined_X_test)
shap.summary_plot(shap_values, combined_X_test,plot_size=[9,6])
# SHAP for positive class
shap.summary_plot(shap_values[1], combined_X_test,plot_size=[9,6])
```

Linear Explainer for Logistic Regression Model

```
explainer = shap.Explainer(descent_model, descent_X_train, feature_perturbation="interventional", model_output="probability")
shap_values = explainer.shap_values(descent_X_test)
shap.summary_plot(shap_values, descent_X_test,plot_size=[9,6])

#Class 1/0
shap.summary_plot(shap_values, descent_X_test,plot_size=[9,6], plot_type='bar')
```