Experimental Summary

Anticipating Micro Chaos in human postural balance
Insights from Stick Balancing

Keshav Bhandari[1],Jianyuan Ni[1], Anne Hee Hiong Ngu[1],Joshua Chang[2],John Milton[3],Yan Yan[1]
[1]Department of Computer Science, Texas State University, USA
[2]Department of Neurology, University of Texas at Austin, USA
[3]Department of Keck Science, Claremont McKenna College, USA

***Abstract*— This report[1] presents a synopsis of an experiment conducted for anticipating micro chaos in human postural balance using deep learning. We base foundation of our experiment on [4, 3]. Preliminary results demonstrate our method achieve nearly perfect accuracy. However, this assessment doesn't justify goodness of our model. Since anticipation of wrongdoing is highly time dependent, we redesign accuracy and other metrics like precision and recall to be time dependent.**

I. DATA

We implemented procedures advocated in [4] to build our dataset. We run the simulation for $10,000$ steps and save simulated data in the disk. We used initial angle randomly from a range between (-5,5) exclusively. We collect $arm\_length(\chi(t))$ and $angle(\Phi(t))$ as our features which we subsequently split into feedback and response set as indicated in Eq.1 and Eq.2.

$$\chi(t) = \begin{cases} feedback(x(t-\tau)), & \text{where } t \in [0, n) \\ response(x(t)), & \text{where } t \in [\tau, n) \end{cases} \quad (1)$$

$$\Phi(t) = \begin{cases} feedback(\phi(t-\tau)), & \text{where } t \in [0, n) \\ response(\phi(t)), & \text{where } t \in [\tau, n) \end{cases} \quad (2)$$

This increase our feature space to another 4 dimensions. We chose discretization steps to be $dt = 0.01\ seconds$. This simulation model simulates 1 data point for every 10 milliseconds, that means at 23rd steps this model has already elapsed 230 milliseconds time and produce 23 data points, which is delay feedback time for our simulator, in terms of time steps delay is 23.

Fig.1 represents data creation process. We repeat this process for 4 features $x(t), x(t-\tau), \phi(t)$ and $\phi(t-\tau)$ and combine, which leads to the dimension of data being $(92, 4)$. We run simulation for $10,000$ iterations for sets of average balance time $\{25.5, 40, 113, 240\}$. Since discretization steps are $0.01 seconds$, this produces an average of $balance\_time/0.01$ samples for each balance time, which are $\{2550, 4000, 11300, 24000\}$ for $\{25.5, 40, 113, 240\}$ respectively. For each simulation, we process data as mention in Fig.1. On average, we will have $(n - \tau - 2w + 1, w)$ number of data for label-0 and label-1 expressed in terms of ordered tuple. Since we know estimated average

[1]This is a preliminary report.

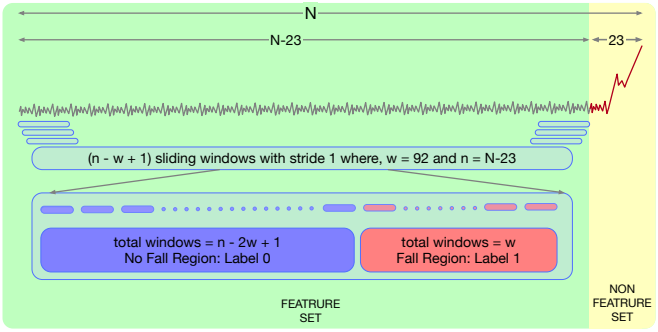

Fig. 1. Visualizing data creation process. We exclude the last section of size 23(signal shown in red color) since we don't want this in our feature space. Feature space is further divided into No-Fall-Region and Fall-Region, which we labelled as 0 and 1. In this way we frame our problem as a binary classification problem. We are asking given a window size of $w = 92$ is there a fall going to happen in next [230 miliseconds,920miliseconds] closed range? Note: here $(n - 2w + 1) >> (w)$, no fall region is significantly larger than fall region.

n, $w = 92$ and $\tau = 23$(in terms of time steps) we can easily compute number of batches. Which leads to $\{(2344, 92), (3794, 92), (11094, 92), (23794, 92)\}$ data size for $\{25.5, 40, 113, 240\}$ respectively, for 1 simulation. We see there is a huge imbalance between label-0 and label-1 in data. To achieve balance, we will run max of each tuple size for each case and balance the data by keeping lesser part. We call this one round of simulations, which includes extra simulations for each balance time. This is actually the difference of max(label-0, label-1) size for each case. So 1 round of simulations includes $\{2253, 3703, 11003, 23703\}$ number of simulation for balancing data in order. This balancing process will make final data size {(2344,2344),(3794,3794),(11094,11094),(23794,23794)} for each balance time in order. We run this process for 10000 rounds, increasing each data size by 10000 times. After combining all balance time data into a single group we will have $(23440000, 37940000, 110940000, 237940000)$ number of batches for each label. This leads to final data size of 2×410260000 batches, which approximate to around 820 $Millioin$ batches(approx 82 $thousands$ per round of simulations). We treat this data as a training data. To achieve a validation and test we will repeat same experiment but we do not balance these sets. This way our validation set will be significantly smaller(approx $41 Million$) than trainnig

set.

II. ARCHITECTURE

Our ideal model should be able to learn the pattern leading to fall(or chaos) accurately. Since our data has periodic randomness, which at the last region(where actual fall took place within 320 to 920 milliseconds) also shows a similar pattern with unnoticeable difference. With this in mind, we want to capture the information more accurately, which we do so by using stacked bidirectional LSTM layers. These layers can learn pattern from both directions, which preserve information from both past and future. We implement fairly simple architecture as shown in Fig-2. To overcome over-fitting due to simulated data(as they have high correlation from one simulation to another) we use a single dropout layer, just after maximally parametrised dense layer. We choose RMSProp as our optimizer, since it prevent gradient exploding and vanishing problem by balancing the gradient. Since we present this problem as a fall-vs-no-fall anticipation, binary-cross-entropy is our default choice for the loss function. We notice that our model converge within few numbers of epochs (12-15); we set large epoch size but adopt early stopping strategy.

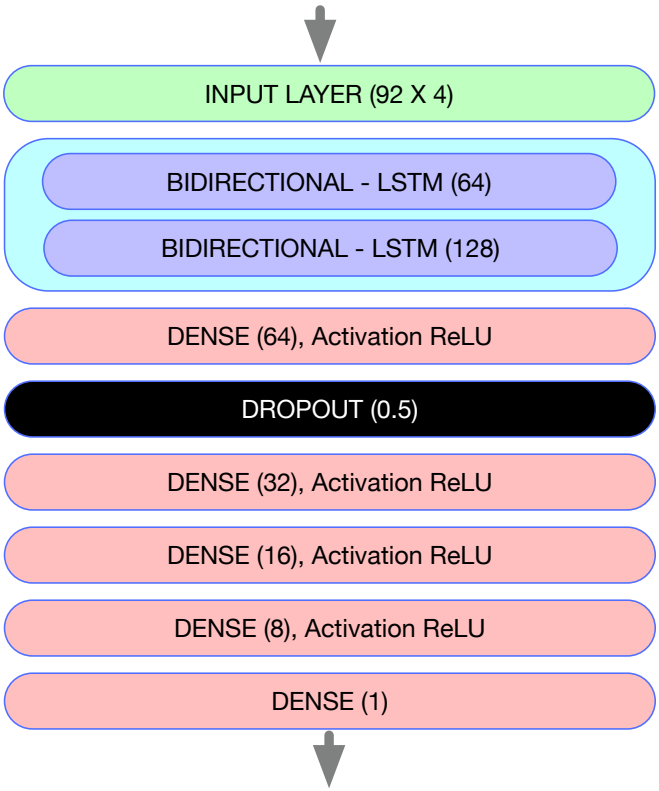

Fig. 2. We adopt fairly simple model with two layers of bidirectional recurrent unit(LSTM) to capture pattern in our data. We further use several layers of dense layer along with a dropout layer to prevent over-fitting.

III. EVALUATION STRATEGY

We want our model to be sensitive and accurate for fall anticipation. We want our model to expect event correctly in terms of time. For example, a terrible model would be a model predicting fall very early where actual fall happens in the distant future. Similarly, another example would be a model predicting no fall where actual fall happens in immediate time step. We want to penalize our measurement metrics heavily over such cases. We consider these aspects and re-model the concept of accuracy, True negative, True positive, False positive and False negative by penalizing these metrics. Based on these ideas, we compute new accuracy, precision, recall and f-ratio.

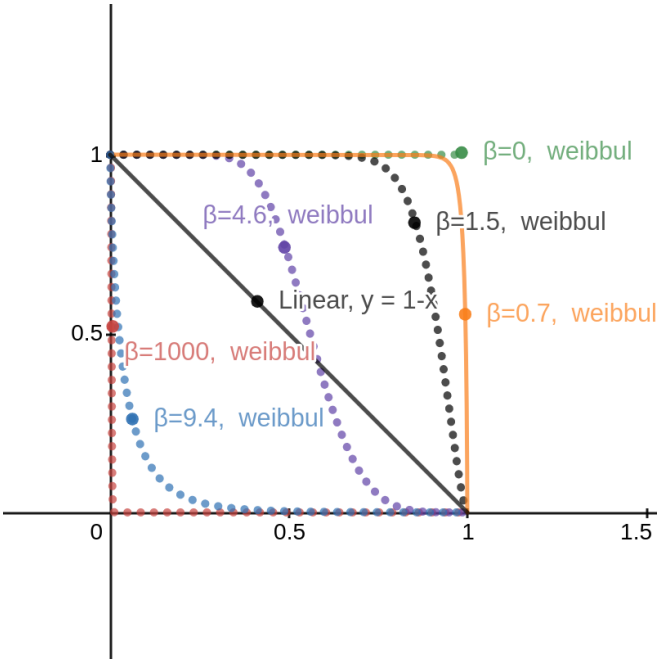

Fig. 3. Showing penalty functions(Weibbul function with different value of β). As we increase the value of β effect of penalty function becomes negligible, so whenever it is large($\beta \to \infty$) metrics behave like ordinary. However, decreasing size of β heavily penalizes the metrics. Note x axis represents time and y axis represents probability. We divide time into (0,1) continuous region by dividing current time step by total time period. From application perspective, instead of using real-time in seconds, we use current window index by total window sizes for each i^th simulation.

We would like to introduce some notations to understand the evaluation strategy. We present our mode as Θ. Similarly i^{th} simulation set as (X_i, Y_i), where (X_i^0, Y_i^0) represent no fall reason and (X_i^1, Y_i^1) represent fall region for i^{th} simulation. Here, $Y_i^0 = \mathbf{0}$ and $Y_i^1 = \mathbf{1}$ for all i in total number of simulations. We define σ as a sigmoid function to map our model output to probability(P). Then,

$$\hat{Y} = \begin{cases} 1 \; if \; P \geq 0.5 \text{ where, } P = \sigma(\Theta(X)) \\ 0 \; else \end{cases} \tag{3}$$

Here, $\hat{Y}_i^0$ corresponds to prediction for no fall region and $\hat{Y}_i^1$ corresponds to fall region for i^{th} prediction. We then compute True Positive Sequence(TP_i), True Negative Sequence(TN_i), False Positive Sequence(FP_i) and False Negative Sequence(FN_i) for i^{th} simulation.

$$TP_i = \begin{cases} 1 \; if \; \hat{Y}_i^1 = Y_i^1 \\ 0 \; else \end{cases} \tag{4}$$

$$TN_i = \begin{cases} 1 \; if \; \hat{Y}_i^0 = Y_i^0 \\ 0 \; else \end{cases} \tag{5}$$

$$\begin{aligned} FP_i &= \mathbf{1} - TN_i \\ FN_i &= \mathbf{1} - TP_i \end{aligned} \tag{6}$$

Based on the work of [2, 1] stick balancing follows Weibbul type function(W) as shown in Eq.7. This survival curve is an approximation of probability for stick not falling given point of time. We expect our model to stick with this nature of Weibbul-curve, that means significantly lower confidence for predicting an event as fall at the beginning of time. So, to measure the practicality of our model, we can use this Weibbul function as penalty function to compute modified metrics(accuracy, precision and recall).

Fig.3 shows plots for different values of β for the weibbul function in Fig.3. From the plot we can see that, as we increase(to very large value) β the curve tries to overlap x and y axis. If we use W with $\beta \to \infty$ and penalize the metrics, then we end up calculating ordinary metrics(accuracy, precision and recall). Similarly, if we choose $\beta \to 0$ then we end up complete penalization of metrics, which is not good. Inspired by the work [2, 1], we choose $\beta = 1.8105$ as optimal value.

$$W = \begin{cases} e^{-(kt)^\beta} \text{ Where, } k = 0.368, \beta = 1.8105 \\ e^{-(kt)^\beta}, \; \beta \to \infty, \text{No Penalty} \\ 1 - x, \text{Linear Penalty} \end{cases} \tag{7}$$

$$\begin{aligned} \omega_{TP_i} &= \sum_{n=0}^{N_i^1} \left((TP)_{i,n} - (FN)_{i,n} \times P_{i,n}^1 \times W(1 - n/N_i^1) \right) \\ \omega_{TN_i} &= \sum_{n=0}^{N_i^0} \left((TN)_{i,n} - (FP)_{i,n} \times P_{i,n}^0 \times W(n/N_i^1) \right) \\ \omega_{FP_i} &= \sum_{n=0}^{N_i^0} \left((FP)_{i,n} + (FP)_{i,n} \times P_{i,n}^0 \times W(n/N_i^1) \right) \\ \omega_{FN_i} &= \sum_{n=0}^{N_i^1} \left((FN)_{i,n} + (FN)_{i,n} \times P_{i,n}^1 \times W(1 - n/N_i^1) \right) \end{aligned} \tag{8}$$

Eq.8 shows computation of weighted TP,TN,FP and FN, denoted as $\omega_{TP}, \omega_{TN}, \omega_{FP}, \omega_{FN}$ respectively. Where, i is i^{th} simulation, N_i is number of windows in i^th simulation, N_i^0, N_i^0 is number of windows in no-fall and fall region in i^th simulation, where $N_i = N_i^0 + N_i^0$. Now, based on Eq.8 we can compute precision and recall as shown in Eq.9, where S represents number of simulation done for testing.

$$\begin{aligned} \omega_{Accuracy} &= \frac{1}{S} \sum_{i=0}^{S} \left(\frac{\omega_{TP_i} + \omega_{TN_i}}{\omega_{TP_i} + \omega_{TN_i} + \omega_{FP_i} + \omega_{FN_i}} \right) \\ \omega_{Precision} &= \frac{1}{S} \sum_{i=0}^{S} \left(\frac{\omega_{TP_i}}{\omega_{TP_i} + \omega_{FP_i}} \right) \\ \omega_{Recall} &= \frac{1}{S} \sum_{i=0}^{S} \left(\frac{\omega_{TP_i}}{\omega_{TP_i} + \omega_{FN_i}} \right) \\ \omega_F &= \frac{1}{S} \sum_{i=0}^{S} \left(\frac{2 \times Precision_i \times Recall_i}{Precision_i + Recall_i} \right) \end{aligned} \tag{9}$$

Metrics	Mode	Precision	Recall	Accuracy	F1-Score
default	with_score	0.85	0.92	0.96	0.89
default	without_score	0.85	0.92	0.96	0.89
linear	with_score	0.78	0.88	0.94	0.83
linear	without_score	0.80	0.92	0.95	0.85
weibbul	with_score	0.76	0.86	0.93	0.81
weibbul	without_score	0.79	0.92	0.95	0.85

TABLE I

Experimental results shown in average for 1000 simulations for each balance time of (25.5s,40s,113s,240s). Note: Mode with and without score refers to penalizing scheme using model confidence(or probability).

IV. RESULTS

We present summary of our experiment in Fig-4,Fig-5 and Table-I. Fig-5 present more condensed information of our experiment. Based on which we can say that penalization with score(or probability that model predicts) on average lower our performance metrics. However, if we consider a perfect model, these scores will be 100%. If we use classical metrics, these metrics won't tell us anything about wrong predictions in different time steps and their severity.

V. CONCLUSION

In this experiment, we implemement mathematical stick balancing model as a source of our data. Our objective is to predict chances of stick fall within next second. Our fairly simple model can easily do this job with impressive accuracy of $> 95\%$. This achievement raises doubts regarding its practicality. We formulate a time based metrics to correctly explain the practicality of the model. These metrics heavily reduce the model performance measure if model does wrong prediction ahead of the time. To our surprise, these metrics are impressive. However, these results are based on random 1000 test simulations.

This is a work in progress and we are excited to try real dataset with our model. If this simulator mimics human stick balancing data fairly, we believe this model will generalise fairly to acceptable range.

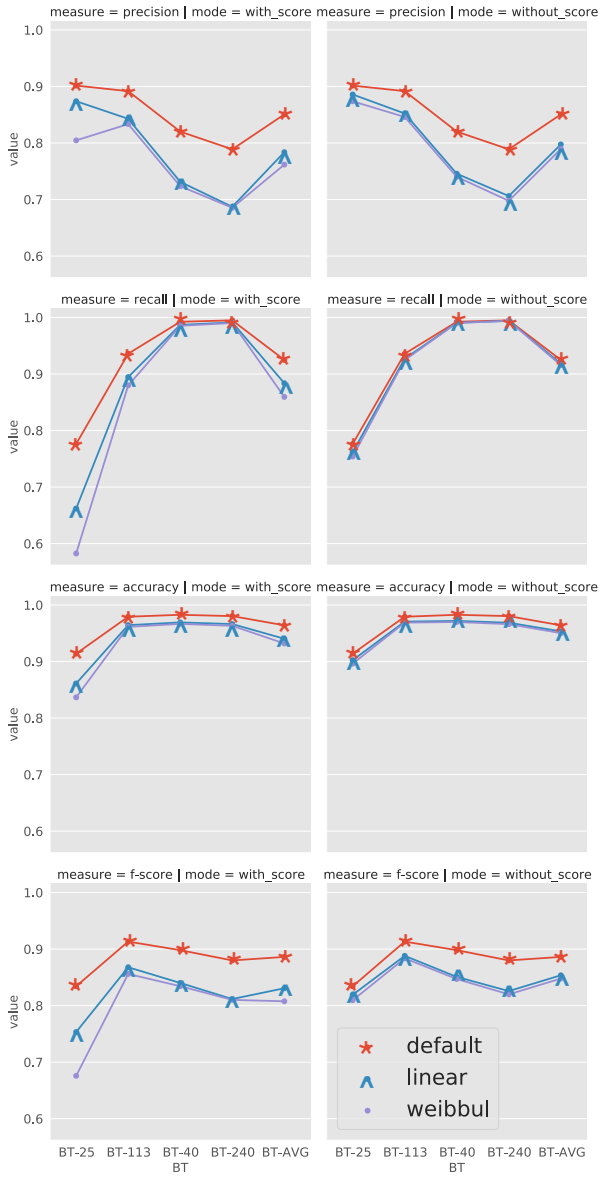

Fig. 4. This plot shows a comparison of metrics used with different penalization settings(linear, Weibbul and none or default). We also show the effect of penalization based using score(probability) vs without using. We can see metrics penalized with Weibbul function is lower compare to linear and non-penalized version. We can also observe that penalization using score lowers the metrics compared with penalization without score.

References

[1] Juan Luis Cabrera, Christian Luciani, and John Milton. "Neural control on multiple time scales: Insights from human stick balancing". In: *Condensed Matter Physics* 945 (Jan. 2006). DOI: 10.5488/CMP.9.2.373.

[2] L. Cabrera1 and J.G. Milton2. "Stick balancing, falls and Dragon-Kings". In: *The European Physical Journal Special Topics* 205 (). URL: https://doi.org/10.1140/epjst/e2012-01573-7.

[3] John G. Milton et al. "Microchaos in human postural balance: Sensory dead zones and sampled time-delayed feedback". In: *Phys. Rev. E* 98 (2 Aug. 2018), p. 022223. DOI: 10.1103/PhysRevE.98.022223. URL: https://link.aps.org/doi/10.1103/PhysRevE.98.022223.

[4] Gabor Stepan, John G. Milton, and Tamas Insperger. "Quantization improves stabilization of dynamical systems with delayed feedback". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 27.11 (2017), p. 114306. DOI: 10.1063/1.5006777. eprint: https://doi.org/10.1063/1.5006777. URL: https://doi.org/10.1063/1.5006777.

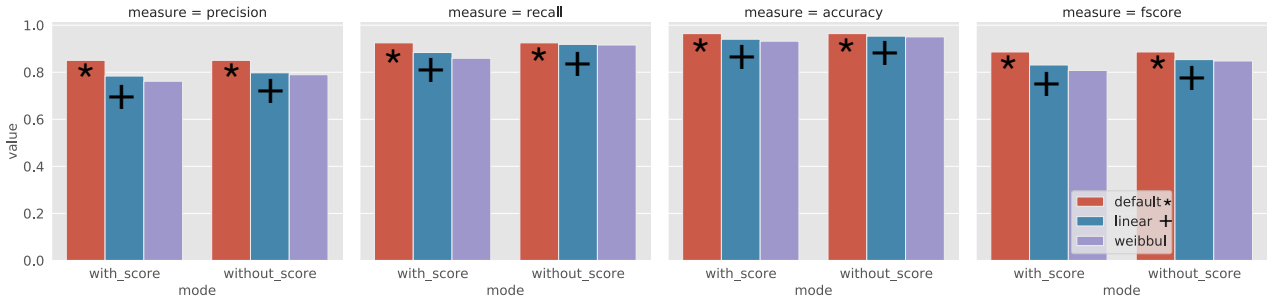

Fig. 5. Comparing metrics shown in Table-I.

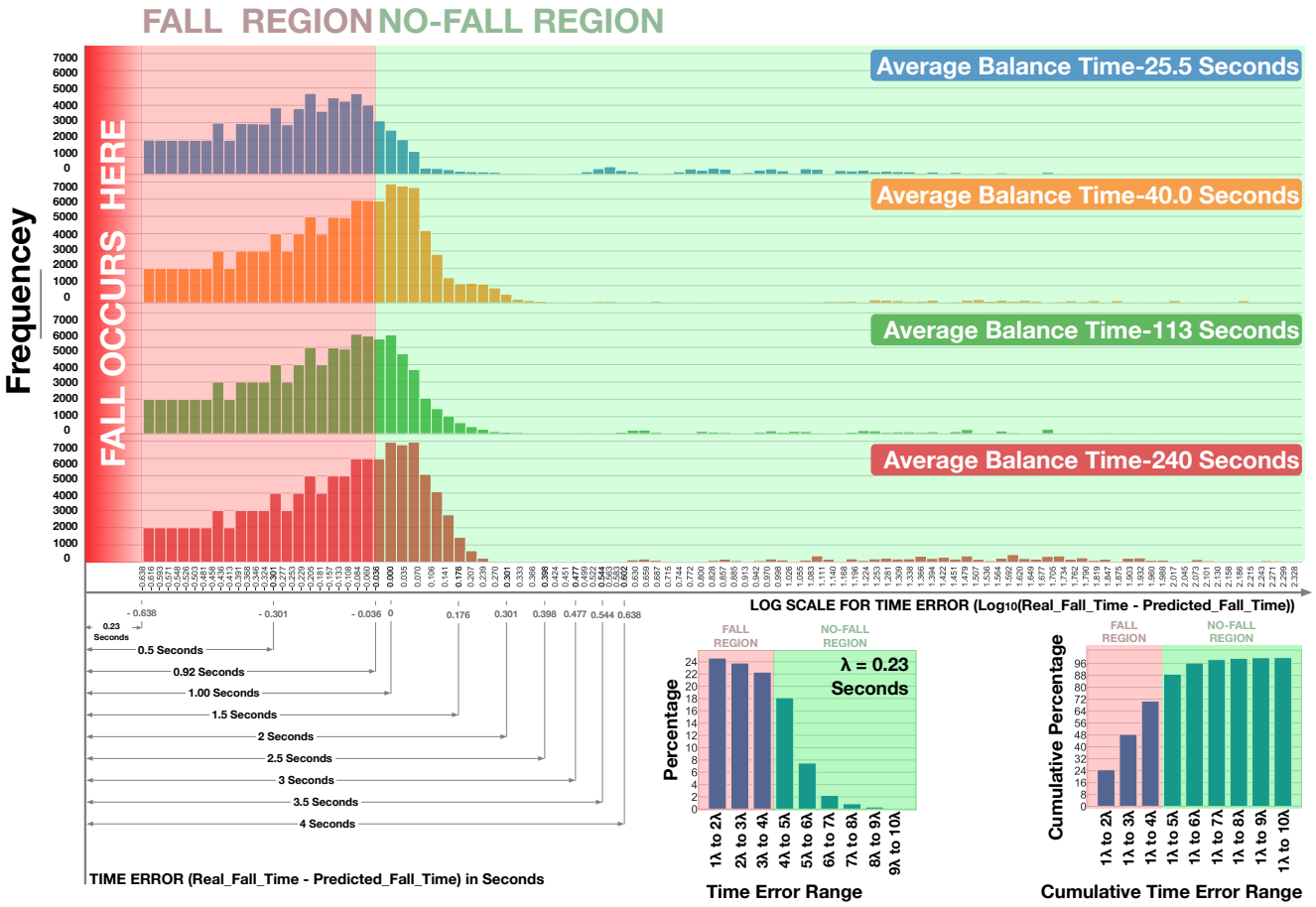

Fig. 6. This plot shows the frequency distribution of time-error defined as (actual_fall_time - predicted_fall_time) for several balance time.Bottom right, first plot contains individual frequency percentage in time-error ranges shown in x-axis and the second plot shows this information cumulatively. Since our model has a prediction confidence for predicting fall in next 0.23(= λ) to 0.92 second range, we should observe higher frequency of error range from (λ *to* 4λ) which is (0.23,0.92) seconds range. We can see this frequency is around 72% of all the time-error range(as shown in bottom right cumulative frequency percentage plot).This suggests significant number of wrong predictions in no-fall region near the border between fall-region and no-fall-region, which is natural. This is based on the design of our experiment, specially related with strategy of defining a no-fall and fall region. Note, time error range(λ *to* 6λ) which is (0.23,1.38) seconds covers almost 96% of all the time-error range(as shown in bottom right figure).

Fig. 7. ROC-AUC Curve for random 100 simulations with average balance time of 25.5 seconds. Note: X-axis represents False Positive Rate and Y-axis represents True Positive Rate, Metrics are penalized using scores(or probability predicted by model).

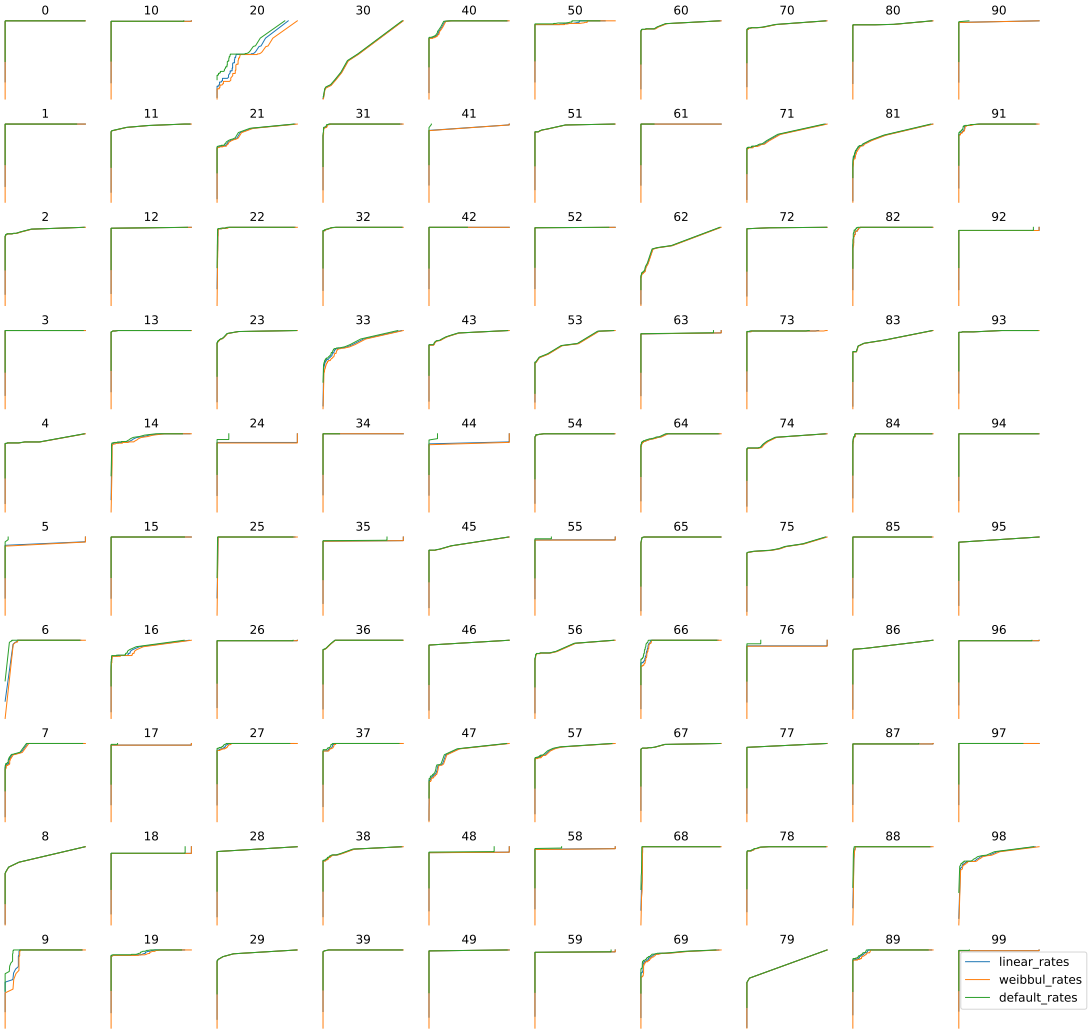

Fig. 8. ROC-AUC Curve for random 100 simulations with average balance time of 25.5 seconds. Note: X-axis represents False Positive Rate and Y-axis represent True Positive Rate.

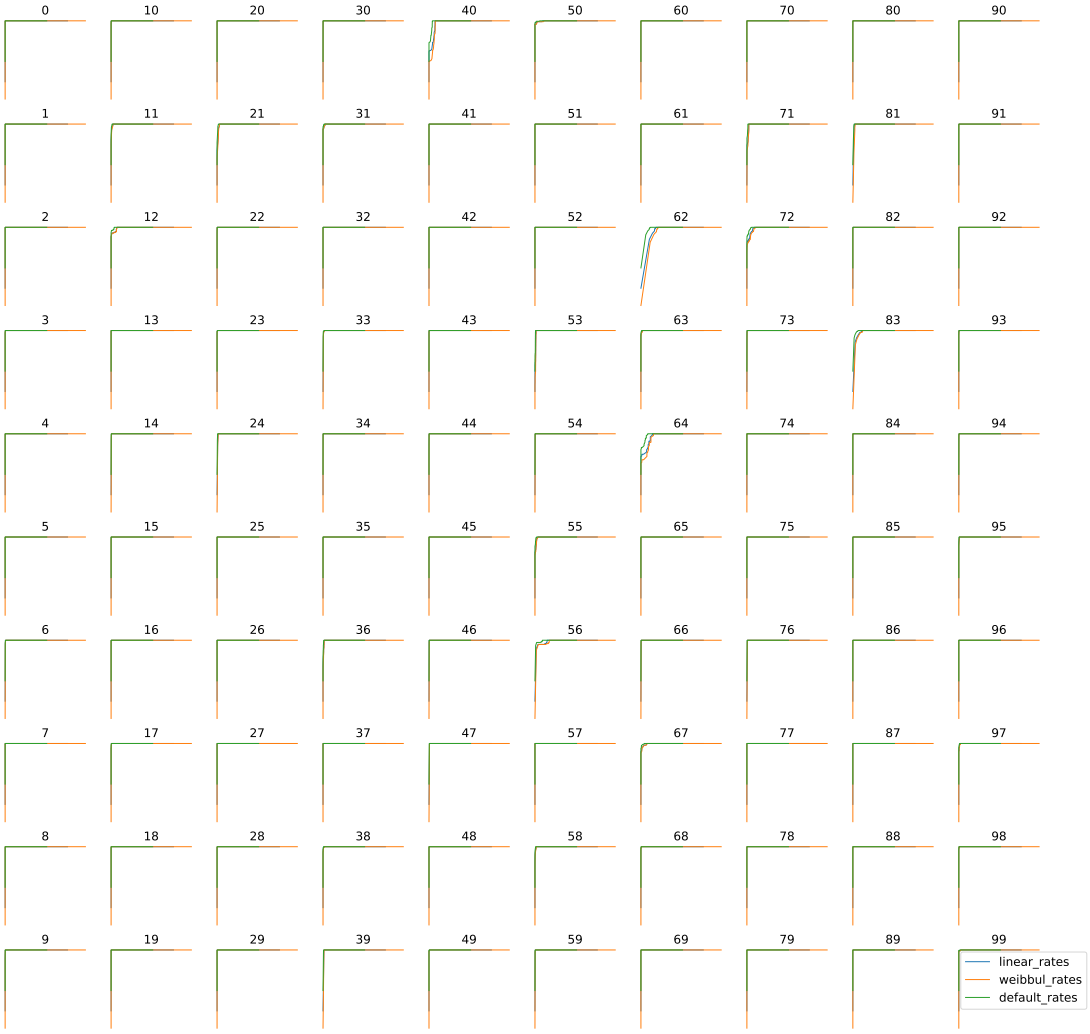

Fig. 9. ROC-AUC Curve for random 100 simulations with average balance time of 40 seconds. Note : X-axis represent False Positive Rate and Y-axis represents True Positive Rate, Metrics are penalized using scores(or probability predicted by model).

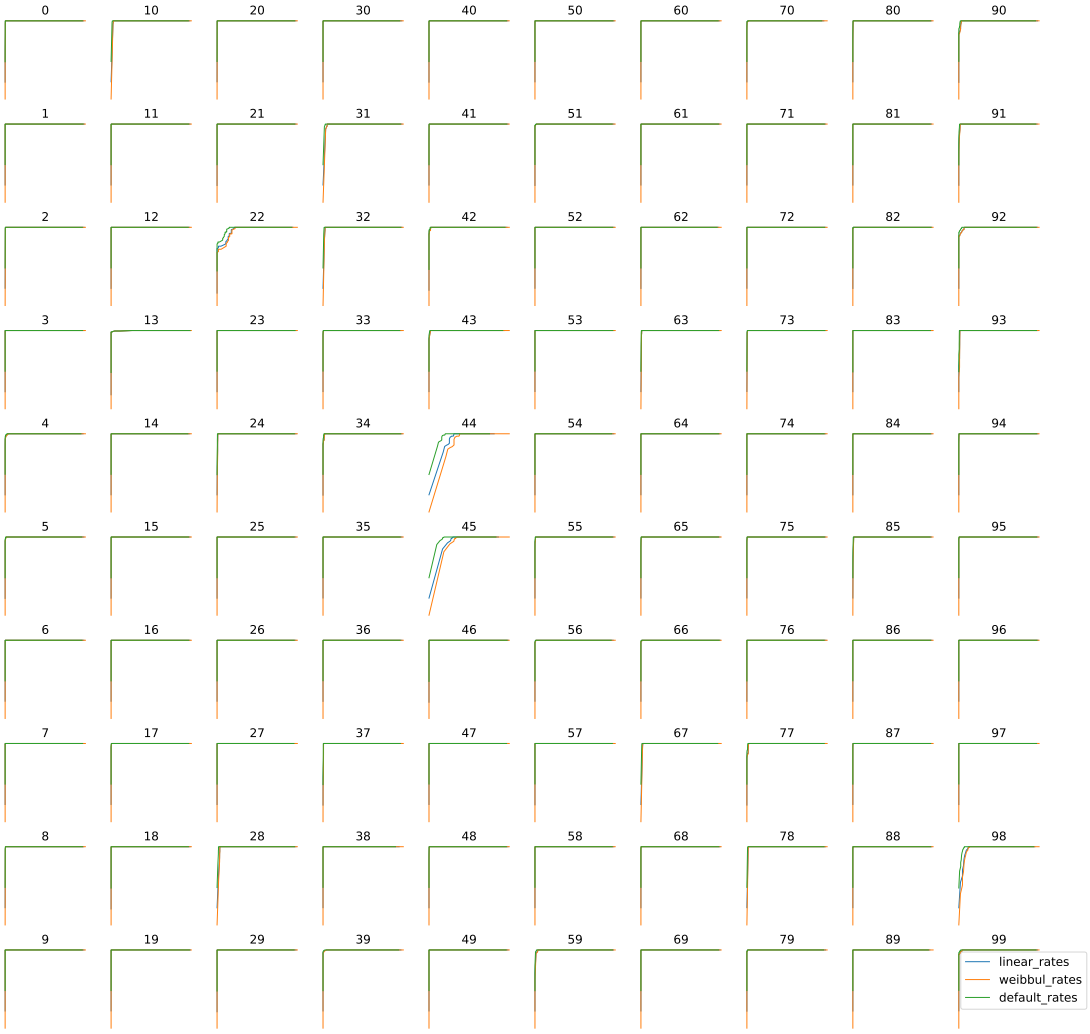

Fig. 10. ROC-AUC Curve for random 100 simulations with average balance time of 40 seconds. Note : X-axis represents False Positive Rate and Y-axis represent True Positive Rate.

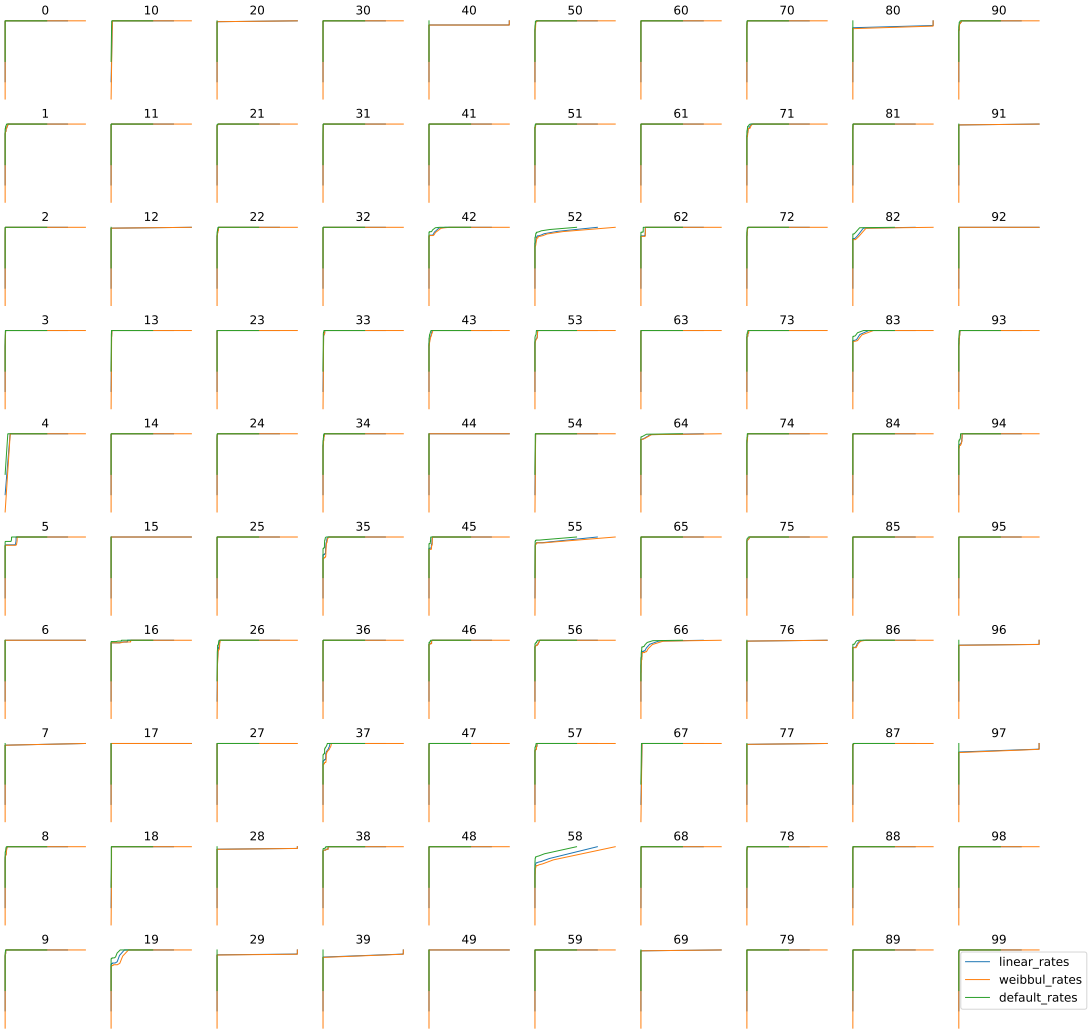

Fig. 11. ROC-AUC Curve for random 100 simulations with average balance time of 113 seconds. Note: X-axis represents False Positive Rate and Y-axis represents True Positive Rate, Metrics are penalized using scores(or probability predicted by model).

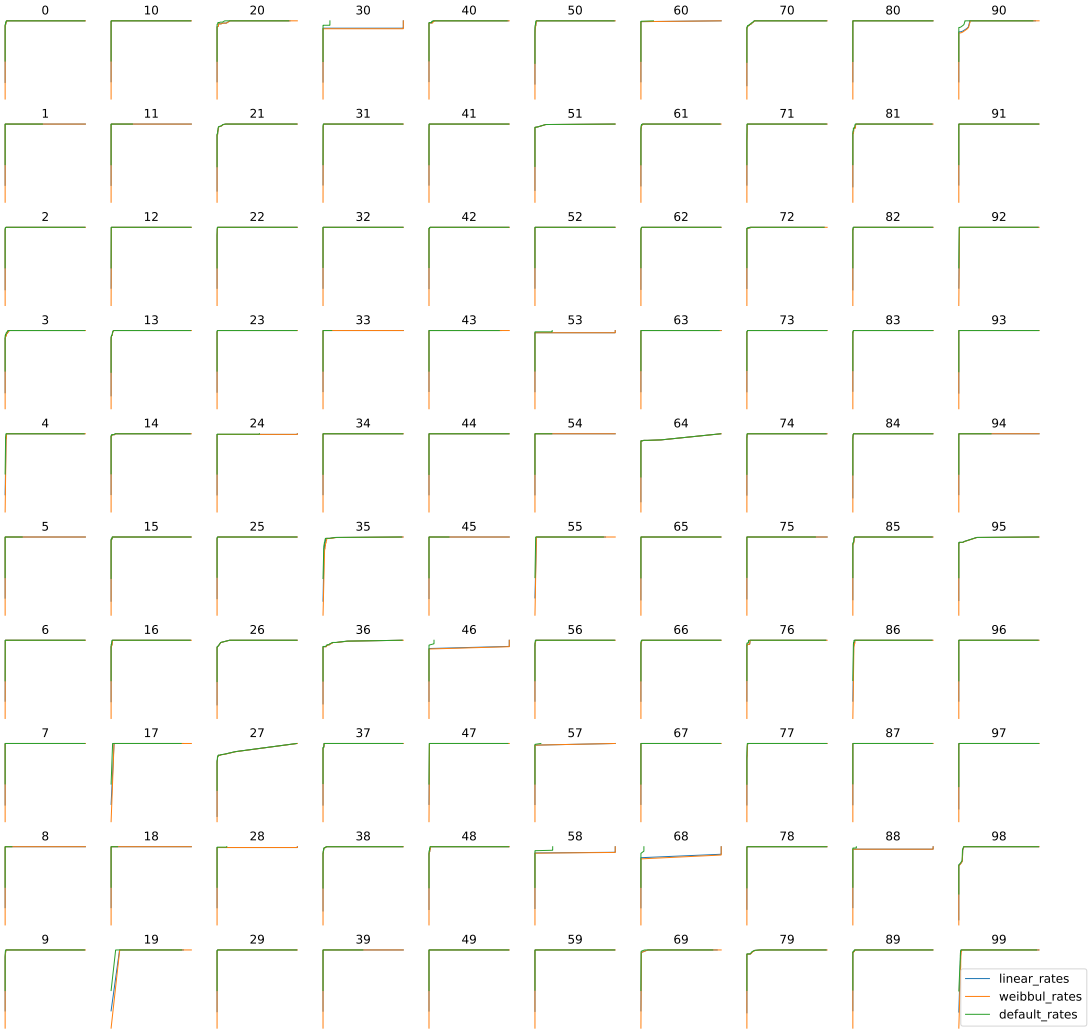

Fig. 12. ROC-AUC Curve for random 100 simulations with average balance time of 113 seconds. Note: X-axis represents False Positive Rate and Y-axis represents True Positive Rate.

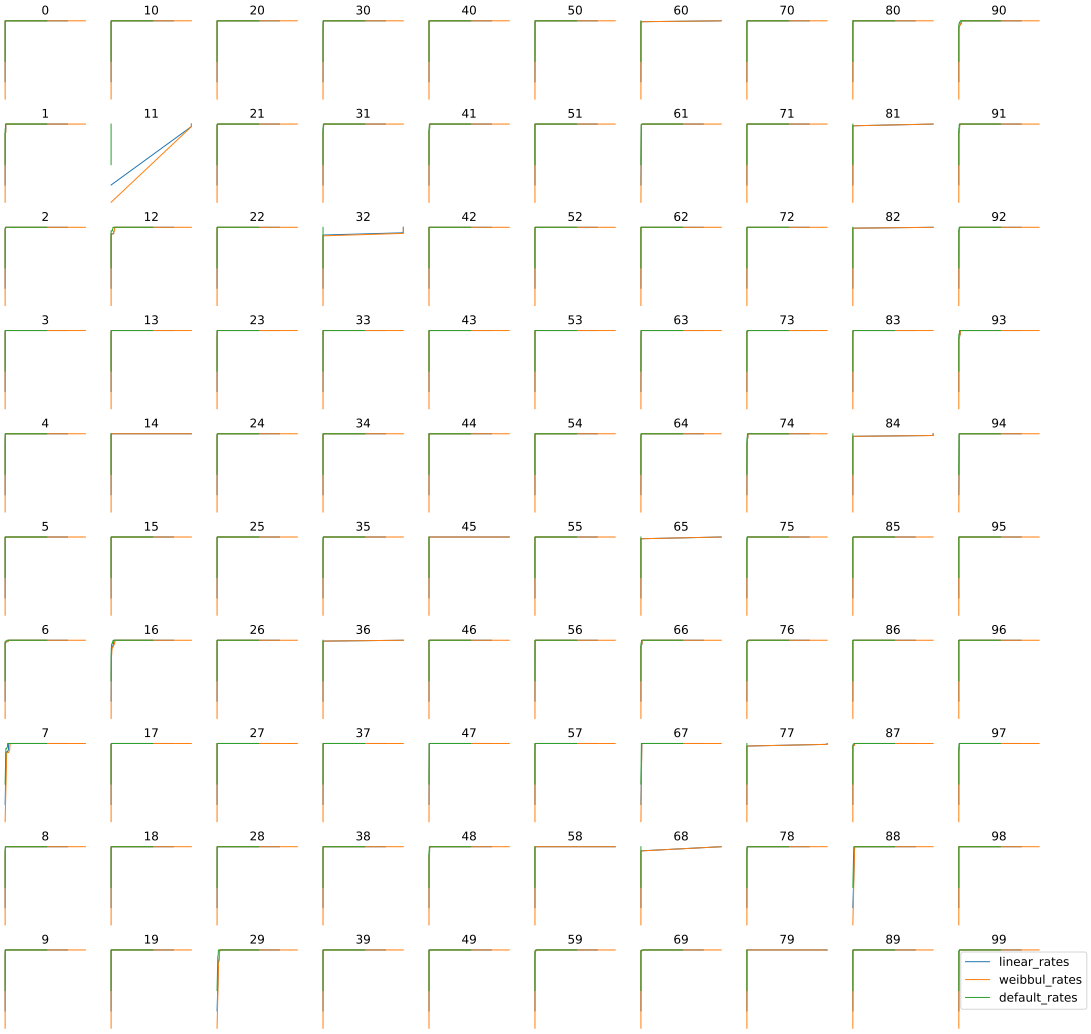

Fig. 13. ROC-AUC Curve for random 100 simulations with average balance time of 240 seconds. Note: X-axis represents False Positive Rate and Y-axis represents True Positive Rate, Metrics are penalized using scores(or probability predicted by model).

Fig. 14. ROC-AUC Curve for random 100 simulations with average balance time of 240 seconds. Note: X-axis represents False Positive Rate and Y-axis represents True Positive Rate.