ANALYZING PLASMA EVOLUTION WITH TRANSFORMERS

GROUP 2315

Supervisor: Anna Braghetto

Members:

- Amirhossein Rostami - 2084299
- Davide Checchia - 2078232
- Jelin Raphael Akkara - 2072064
- Kiamehr Javid - 2084294

OUTLINE

I. PROBLEM AND APPROACH OVERVIEW

II. DATA PREPARATION

III. VALIDATION AMONG MODEL ARCHITECTURES

IV. IMPROVEMENT OF BEST MODEL

V. CONCLUSION

I.
PROBLEM AND APPROACH OVERVIEW

1. Introduction to Problem
2. Transformers
3. General Approach
4. Challenges Faced
5. Methods Used

1. INTRODUCTION TO THE PROBLEM

- Plasma is confined in a torus, in a helical shape, using strong magnetic fields. Magnetic instabilities can result in magnetic reconnection (crash), which is a current issue in plasma physics.

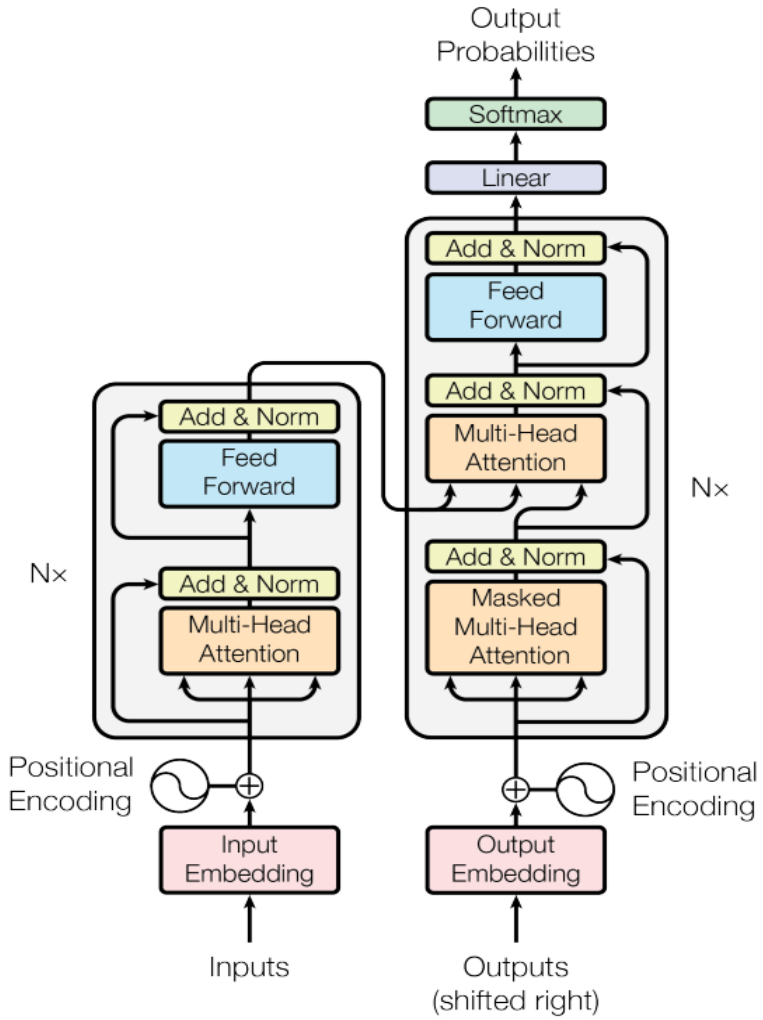
- We are interested in the prediction of crashes, based on the features (e.g. magnetic field instabilities, safety factors) of the system.

- From the machine learning perspective, this translates to a binary classification problem with timeseries data. Using a time window of feature values, we need to predict if a crash is going to occur in a given time period.

- The instances are time-series of the values of the magnetic instabilities at different positions of a torus. The saw_crash labels simply indicate the presence or absence of a crash.

2. TRANSFORMERS

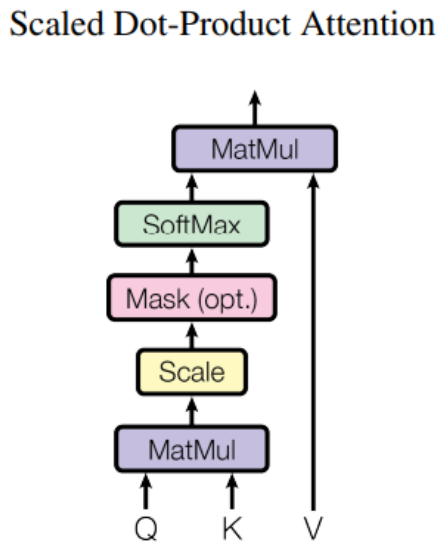
- A sequence-to-sequence architecture: Transforms one sequence to another
- Extensively used in Natural Language Processing, Timeseries Evaluation, etc.
- Implements **Attention mechanism**: This looks at an input sequence and assigns importance (attention) to each section of the sequence.
- Consists of two main components: **Encoder** (takes input and implements self-attention) and **Decoder** (takes in output samples and implements masked attention).



2. TRANSFORMERS – SCALED DOT PRODUCT (SELF) ATTENTION

Three Representations: Each word embedding is projected onto three different spaces, obtaining three different vectors:

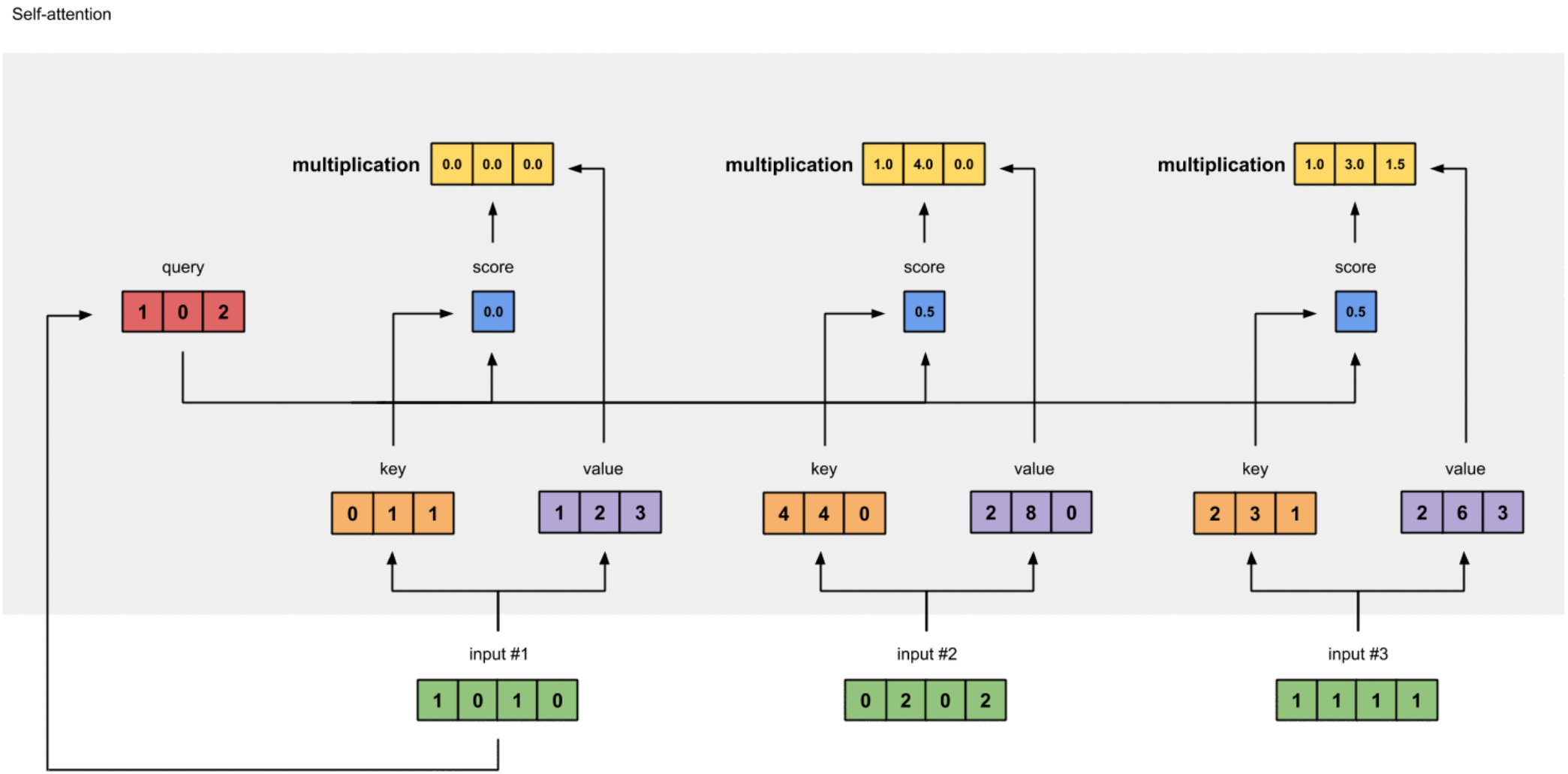
- **The Query vector:** Captures the current word's context and is used to compare for relevant context in other words of the sentence.
- **The Key vector:** Contains summarized information of all words in the sentence, and is used as a reference to find the relevant words to the current word.
- **The Value vector:** Provides the actual word, with little modification to its general meaning. The attention scores are used to weight these value vectors in the end.



Attention Score: The query vector is compared with all the key vectors, and an attention score is generated. We use the scaled dot-product attention:

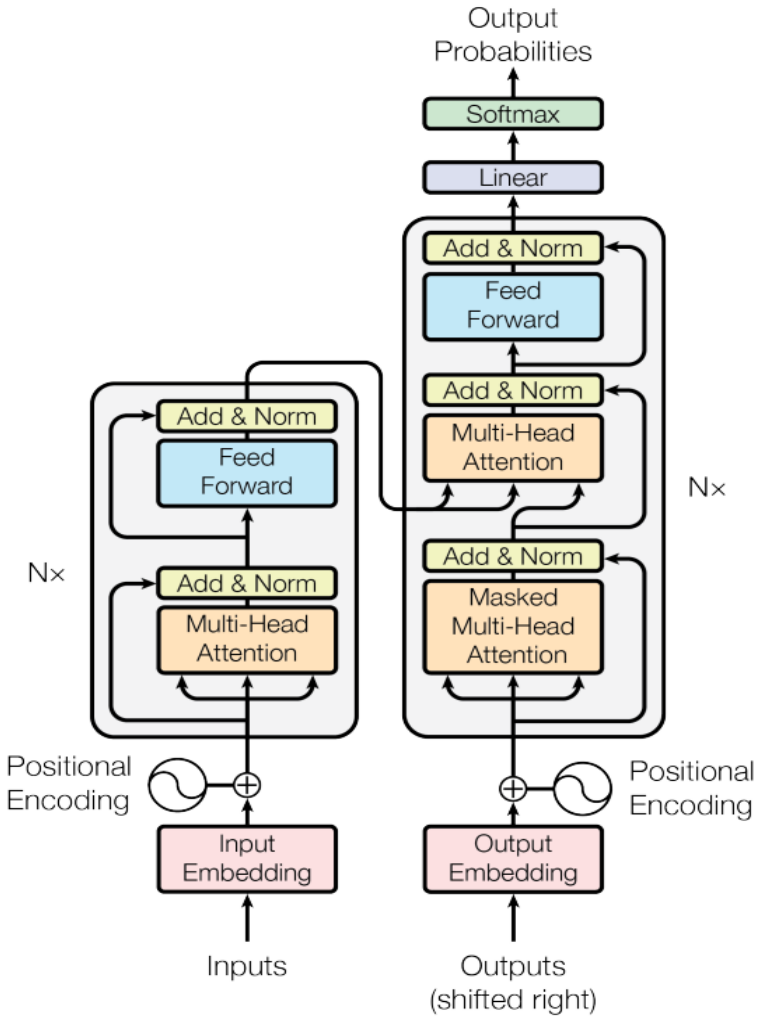
$$Attention(Q, K, V) = softmax\left(\frac{Q\ K^T}{\sqrt{d}}\right)V$$

2. TRANSFORMERS – SCALED DOT PRODUCT ATTENTION



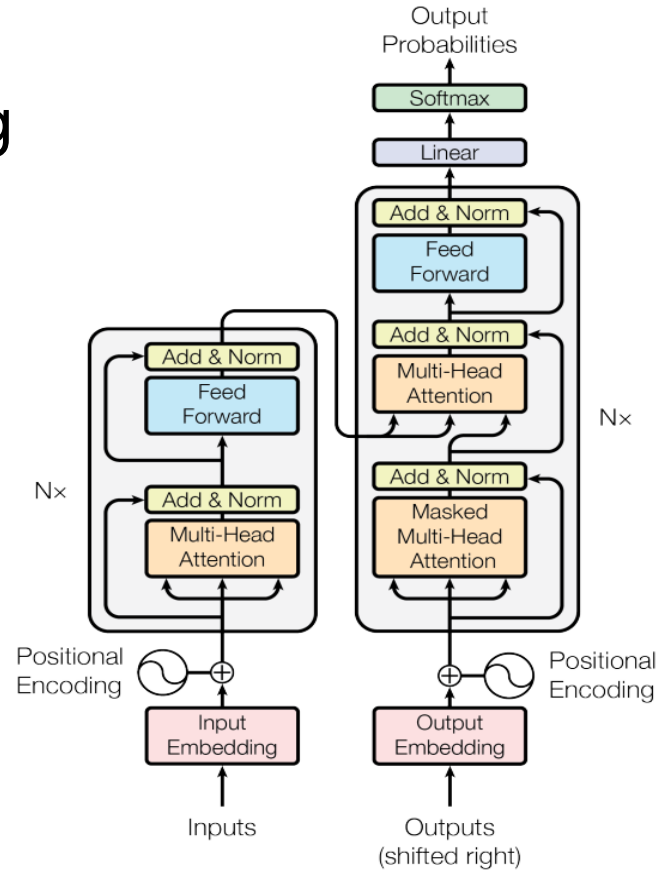
2. TRANSFORMERS – Natural Language Processing

- **Input Processing:** Input sentence is received, **tokenized** (marking each word's position in the input) and converted to **embedded vectors** (mapping the words to an embedding space to give them general context).
- **Encoder:**
  - **Multi-Head Attention:** Applies attention to itself (self-attention) using different sets of weight matrices. The different sets define the multiple heads.
  - **Feed Forward Network:** Learns patterns using the attended and improved embeddings.
  - **Add and Normalization:** Adds the input to the result so as to maintain initial meaning or context. Normalizes the results so that the model works with equally scaled variables.



2. TRANSFORMERS – Natural Language Processing

- **Decoder:**
  - Auto-regressive
  - **Masked Multi-Head Attention:** Applying self attention by masking future words. This ensures that all learning and predictions happen using current and previous data.
  - **Second Multi-Head Attention:** The encoder's output is used as the query and key, and the first multi-head attention's output is the value.
  - **Output:** Provides probability scores over vocabulary



SCALED SCORES			
0.7	0.1	0.1	0.1
0.1	0.6	0.2	0.1
0.1	0.3	0.6	0.1
0.1	0.3	0.3	0.3

*

LOOK-AHEAD MASK			
1	-inf	-inf	-inf
1	1	-inf	-inf
1	1	1	-inf
1	1	1	1

=

MASKED SCORES			
0.7	-inf	-inf	-inf
0.1	0.6	-inf	-inf
0.1	0.3	0.6	-inf
0.1	0.3	0.3	0.3

2. TRANSFORMERS – TIME SERIES DATA

- In Timeseries data:
  - Sentence -> A selected time window of evolution of features (matrix of shape: (Time window, Row length))
  - Word -> Set of features at one time instance (each row)

- We use only Encoder here since we are dealing with a binary classification task

Sentence

Word

Unnamed: 0	Time	F	saw_crash	bz_m1n6_a	bz_m1n7_a	bz_m1n8_a	bz_m1n11_a	...	br_m1n20_02	q_0	q_01	q_02
0	1000.0	-0.143200	0.0	0.000024	0.000488	0.001192	0.002039	...	0.000071	0.151312	0.147824	0.144317
1	1010.0	-0.142802	0.0	0.000024	0.000480	0.001185	0.002045	...	0.000070	0.151268	0.147823	0.144318
2	1020.0	-0.142410	0.0	0.000025	0.000473	0.001179	0.002050	...	0.000069	0.151213	0.147824	0.144317
3	1030.0	-0.142024	0.0	0.000024	0.000466	0.001173	0.002056	...	0.000068	0.151149	0.147827	0.144315
4	1040.0	-0.141641	0.0	0.000024	0.000461	0.001167	0.002062	...	0.000066	0.151085	0.147830	0.144314

3. GENERAL APPROACH

PREPARING DATA

- Cleaning, Filtering Data
- Splitting into testing and training sets

DEFINE METRICS

- Accuracy
- Recall
- Precision
- F-Beta Score

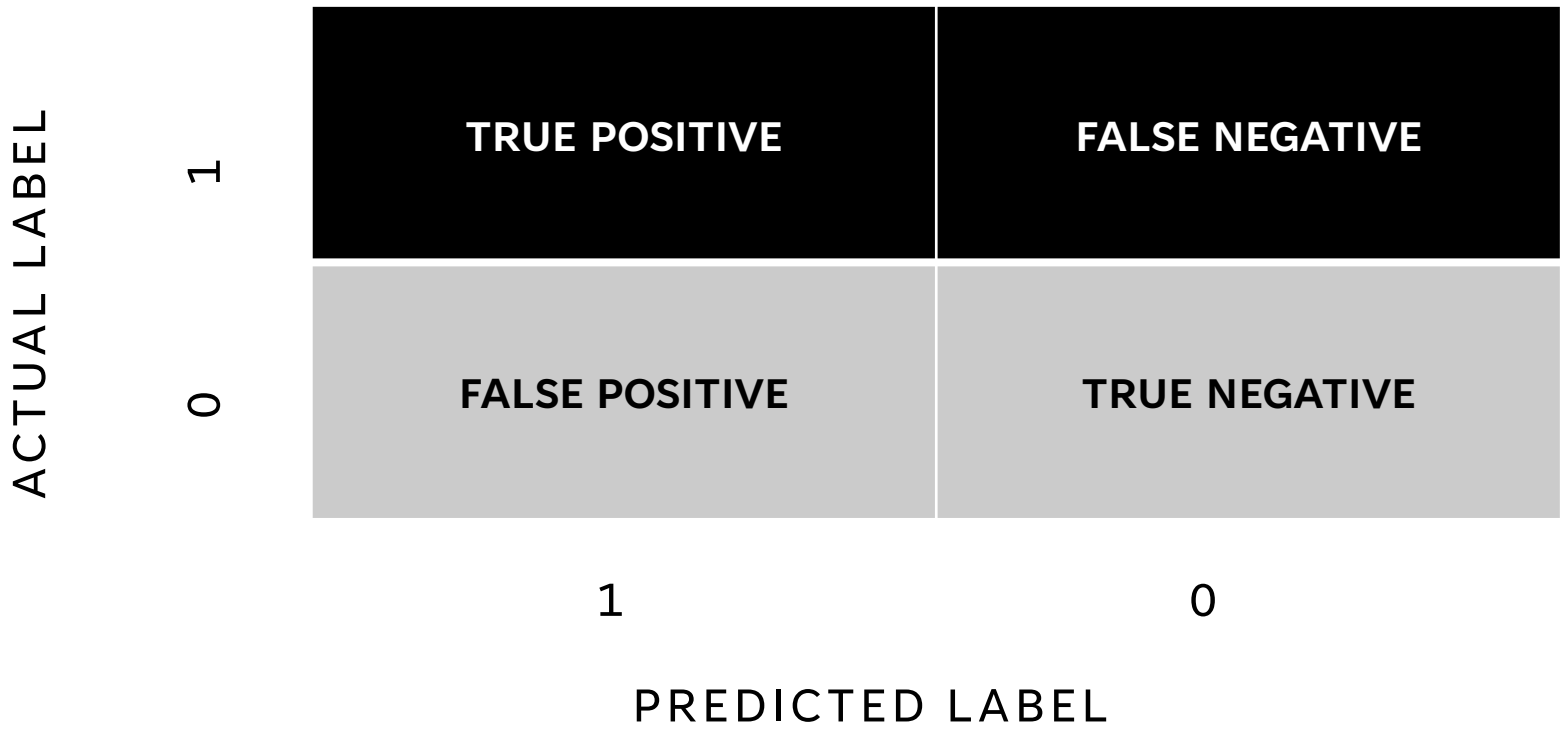
DEFINE MODELS

- Convolutional Neural Network
- Dense Neural Network
- Transformers

VALIDATION AND GRID-SEARCH

IMPROVING BEST MODEL

3. GENERAL APPROACH – CONFUSION MATRIX



- **True Positive:** Predicting crash when actual crash is present.
- **False Negative:** Predicting no crash when actual crash is present.
- **False Positive:** Predicting crash when no actual crash is present.
- **True Negative:** Predicting no crash when no crash is present.

3. GENERAL APPROACH – METRICS

RECALL

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

- Used when cost of false negatives is high. A higher recall indicates fewer false negatives.
- False Negative is when the model predicts zero when it is actually one, i.e. it predicts no crash when an actual crash is present.

PRECISION

$$Precision\ = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

- Used when the cost of false positives is high. This tells us how good the model is in predicting crashes but also checks if it predicts a crash even when there is none
- False positive is when the model predicts one when it is actually zero, i.e. it predicts a crash when there is not one.

F-BETA SCORE:

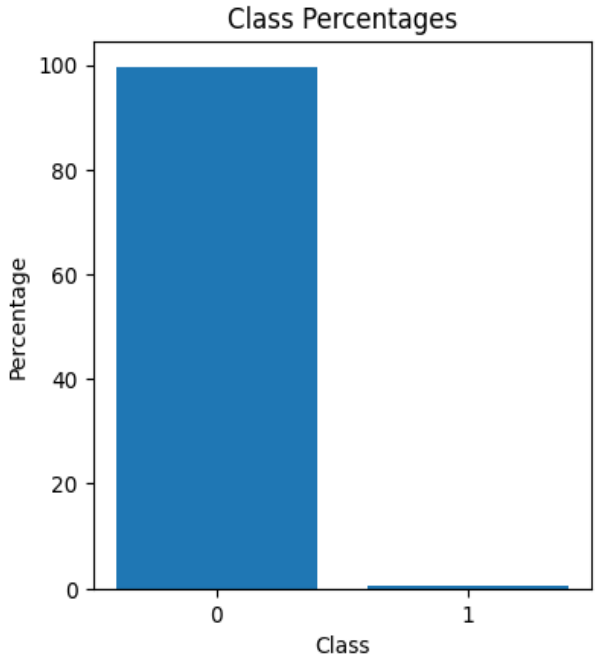
- Weighted Harmonic Mean between Precision and Recall, with a weight factor (beta) that assigns importance to recall

$$(1 + \beta^2)\ \frac{Precision\ * Recall}{(\beta^2\ * Precision) + Recall}$$

4. CHALLENGES FACED

DATA IMBALANCE

- Large percentage of zeros as compared to ones
- Percentages: (0: 99.5% , 1: 0.48%)



DEAD AND UNSTABLE SOLUTIONS

- Some solutions kept predicting either 1 or zero all the time
- Unreliable training, gets very different results each time

COMPUTATIONAL COSTS

- Grid Search on all possible combinations not possible

5. METHODS USED

PRINCIPAL COMPONENT ANALYSIS

STRATIFICATION

DOWN-SAMPLING OF MAJORITY CLASS

SPARSE GRID SEARCH

REPEATED ANALYSES

II.
DATA PREPARATION

1. Structure of Data
2. Preparing Data
3. MP vs MP-Less Analysis

1. STRUCTURE OF DATA

FILES

- **MP Data**: With Magnetic Perturbation.
  - Rows: ~40000
  - Columns: 67
- **MP-Less Data:** Without Magnetic Perturbation
  - Rows: ~40000
  - Columns: 67

COLUMNS

- **Time stamp:** Measured in Alfven units
- **F parameter:** Reversal parameter, sudden fall near crash
- **Saw_crash:** Binary label for crash
- **Toroidal Magnetic Field Instability Amplitude:** 15*3 columns, each set measured at different radial positions: r=(0, 0.1a, 0.2a)
- **Safety Factors:** 3 columns at three different radial positions

2. PREPARING THE DATA – METHODS USED

TRAINING AND PREDICTION WINDOWS

- Training Window (L): Length of window used for training
- Prediction Time (dt): How long in the future we want to check for a crash
- Prediction Window (WR): Checking for a crash inside window of length WR at time dt

STRATIFICATION

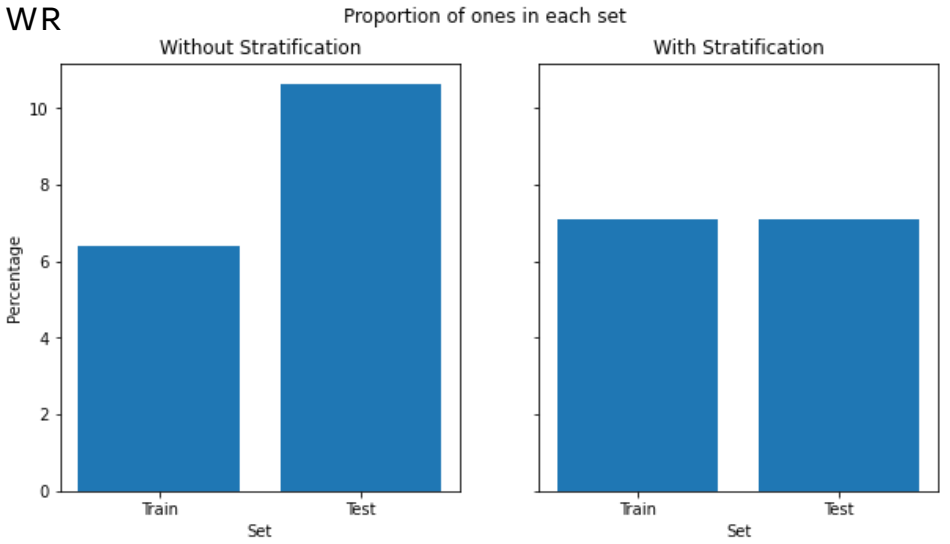
- Maintaining equal proportion of ones when splitting the data into two sets

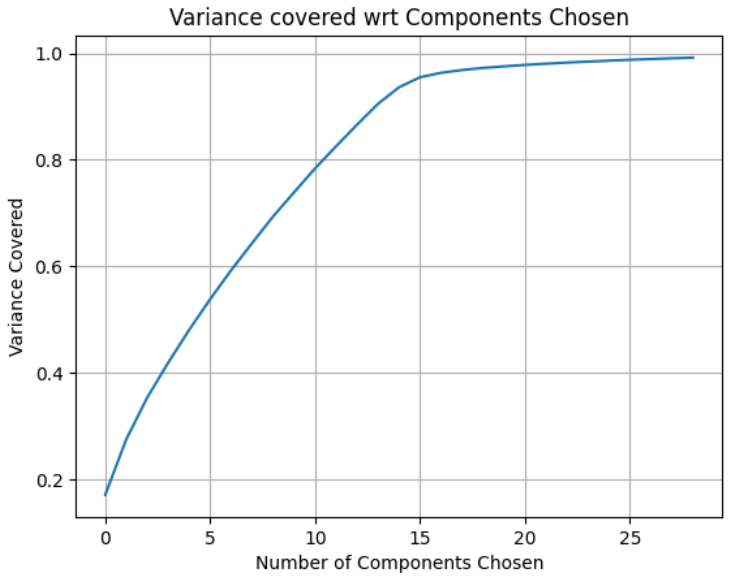
DOWN-SAMPLING OF ZEROS

- Randomly choosing zeros with an acceptance rate of 0.2

PRINCIPAL COMPONENT ANALYSIS

- Chose number of components such that 99% variance is explained: pca_num = 29





2. PREPARING THE DATA – DATA SNIPPET

(L: 6, DT: 4, WR: 0)

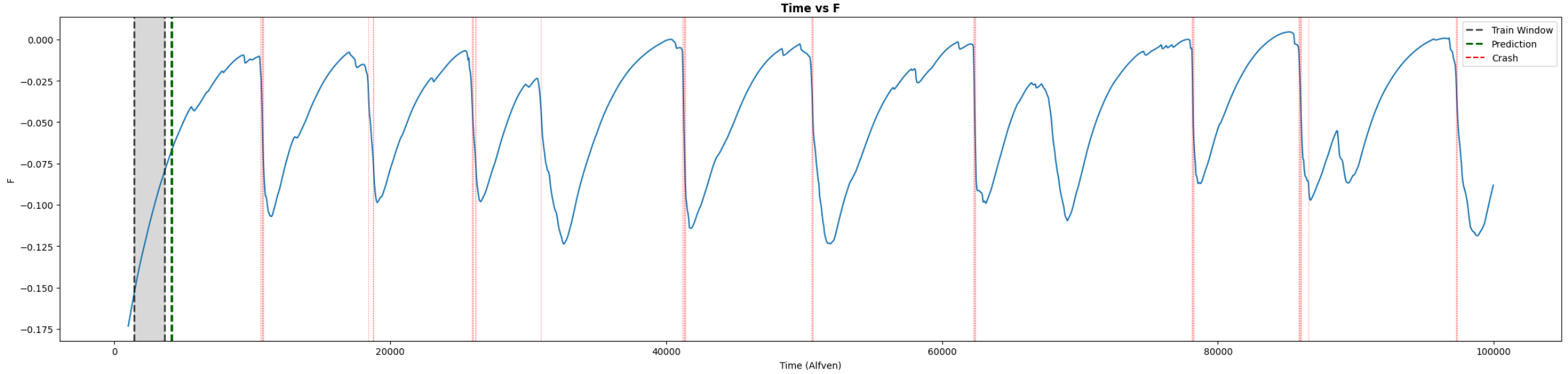
	Unnamed: 0	Time	F	saw_crash	bz_m1n6_a	bz_m1n7_a	bz_m1n8_a	bz_m1n9_a	bz_m1n10_a	bz_m1n11_a	...	br_m1n20_02	q_0	q_01	q_02
0	0	1000.0	-0.143200	0.0	0.000024	0.000488	0.001192	0.001451	0.002297	0.002039	...	0.000071	0.151312	0.147824	0.144317
1	1	1010.0	-0.142802	0.0	0.000024	0.000480	0.001185	0.001452	0.002281	0.002045	...	0.000070	0.151268	0.147823	0.144318
2	2	1020.0	-0.142410	0.0	0.000025	0.000473	0.001179	0.001453	0.002266	0.002050	...	0.000069	0.151213	0.147824	0.144317
3	3	1030.0	-0.142024	0.0	0.000024	0.000466	0.001173	0.001453	0.002251	0.002056	...	0.000068	0.151149	0.147827	0.144315
4	4	1040.0	-0.141641	0.0	0.000024	0.000461	0.001167	0.001453	0.002237	0.002062	...	0.000066	0.151085	0.147830	0.144314
5	5	1050.0	-0.141263	0.0	0.000024	0.000456	0.001160	0.001453	0.002223	0.002067	...	0.000063	0.151023	0.147831	0.144314
6	6	1060.0	-0.140889	0.0	0.000024	0.000452	0.001153	0.001453	0.002209	0.002072	...	0.000060	0.150968	0.147831	0.144314
7	7	1070.0	-0.140518	0.0	0.000024	0.000448	0.001145	0.001451	0.002195	0.002076	...	0.000057	0.150916	0.147831	0.144313
8	8	1080.0	-0.140150	0.0	0.000024	0.000445	0.001137	0.001449	0.002181	0.002080	...	0.000054	0.150867	0.147832	0.144311
9	9	1090.0	-0.139785	0.0	0.000025	0.000442	0.001129	0.001447	0.002167	0.002084	...	0.000051	0.150819	0.147834	0.144308

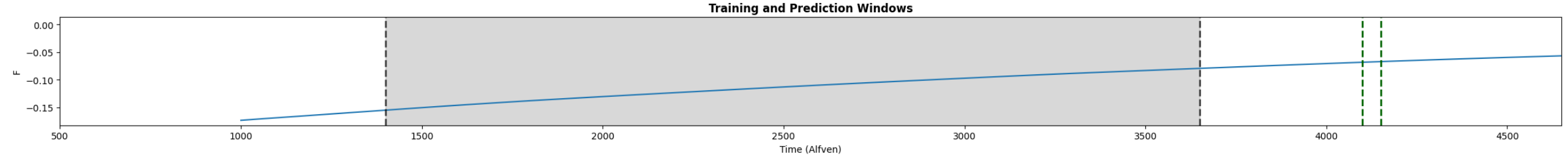
2. PREPARING THE DATA – DATA SNIPPET

(L: 6, DT: 4, WR: 1)

	Unnamed: 0	Time	F	saw_crash	bz_m1n6_a	bz_m1n7_a	bz_m1n8_a	bz_m1n9_a	bz_m1n10_a	bz_m1n11_a	...	br_m1n20_02	q_0	q_01	q_02
0	0	1000.0	-0.143200	0.0	0.000024	0.000488	0.001192	0.001451	0.002297	0.002039	...	0.000071	0.151312	0.147824	0.144317
1	1	1010.0	-0.142802	0.0	0.000024	0.000480	0.001185	0.001452	0.002281	0.002045	...	0.000070	0.151268	0.147823	0.144318
2	2	1020.0	-0.142410	0.0	0.000025	0.000473	0.001179	0.001453	0.002266	0.002050	...	0.000069	0.151213	0.147824	0.144317
3	3	1030.0	-0.142024	0.0	0.000024	0.000466	0.001173	0.001453	0.002251	0.002056	...	0.000068	0.151149	0.147827	0.144315
4	4	1040.0	-0.141641	0.0	0.000024	0.000461	0.001167	0.001453	0.002237	0.002062	...	0.000066	0.151085	0.147830	0.144314
5	5	1050.0	-0.141263	0.0	0.000024	0.000456	0.001160	0.001453	0.002223	0.002067	...	0.000063	0.151023	0.147831	0.144314
6	6	1060.0	-0.140889	0.0	0.000024	0.000452	0.001153	0.001453	0.002209	0.002072	...	0.000060	0.150968	0.147831	0.144314
7	7	1070.0	-0.140518	0.0	0.000024	0.000448	0.001145	0.001451	0.002195	0.002076	...	0.000057	0.150916	0.147831	0.144313
8	8	1080.0	-0.140150	0.0	0.000024	0.000445	0.001137	0.001449	0.002181	0.002080	...	0.000054	0.150867	0.147832	0.144311
9	9	1090.0	-0.139785	0.0	0.000025	0.000442	0.001129	0.001447	0.002167	0.002084	...	0.000051	0.150819	0.147834	0.144308

2. PREPARING THE DATA – METHODS USED





3. MP VS MP-LESS ANALYSIS

- **Two types** of datasets
  - With magnetic perturbation
  - Without magnetic perturbation
- The model with the best parameters is trained on each set of data and tested independently.
- The results are not as satisfying compared to the trainings on the merged dataset.

Magnetic Perturbation	Applied	Not Applied
Accuracy	77.6 %	73.1 %
Recall	88.9 %	86.8 %
Precision	9.3%	6.5 %

III.
VALIDATION AMONG MODEL ARCHITECTURES

1. Schema of Model Architectures
2. Grid Search on Models
3. Why Transformers
4. Plots and Results

1. SCHEMA OF MODEL ARCHITECTURES

DATA PARAMETERS: (L: 30, DT: 10, WR: 0, PCA: FALSE)

MODEL 1: CNN

- Convolutional layer
- Dropout
- Batch Normalization
- Repeat for num_layers
- Output Dense layer

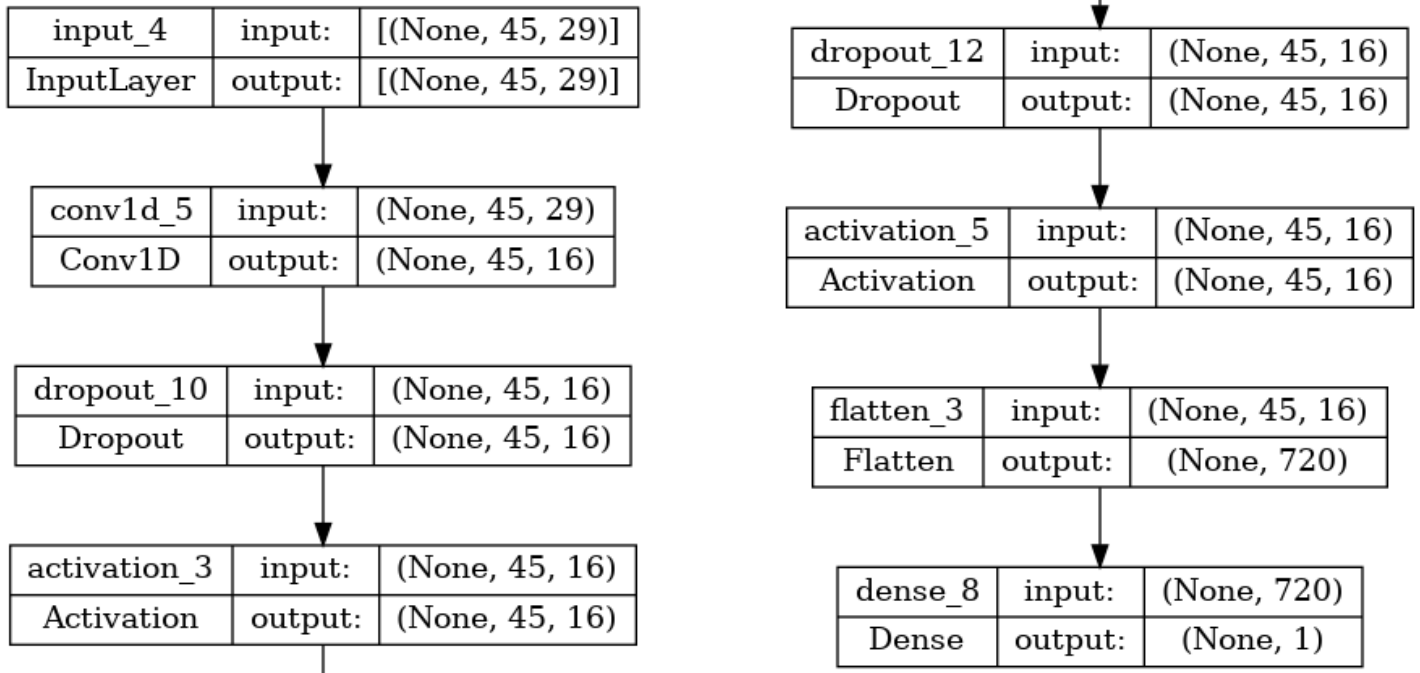
MODEL 2: DNN

- Dense layers of sizes from 8 till 2048.
- Dropout
- Output Dense layer

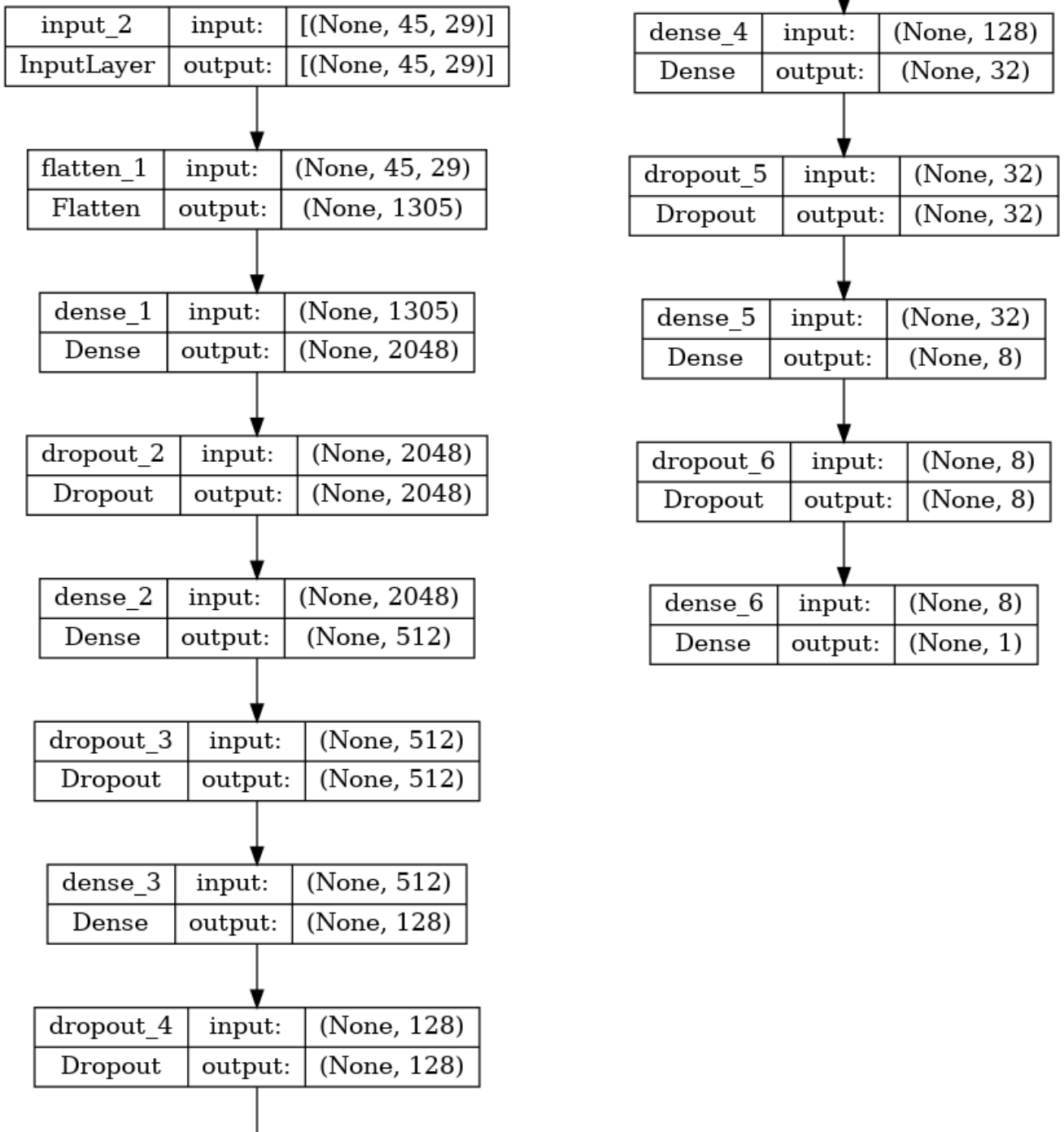
MODEL 3: TRANSFORMER

- Encoder (N times)
  - Layer Normalization
  - MultiHead Attention
  - Dropout
  - Residual Connection
  - Layer Normalization
  - 1D Convolutional Layer
  - Dropout
  - 1D Convolutional Layer
  - Residual Connection
- Possible Dense Layers

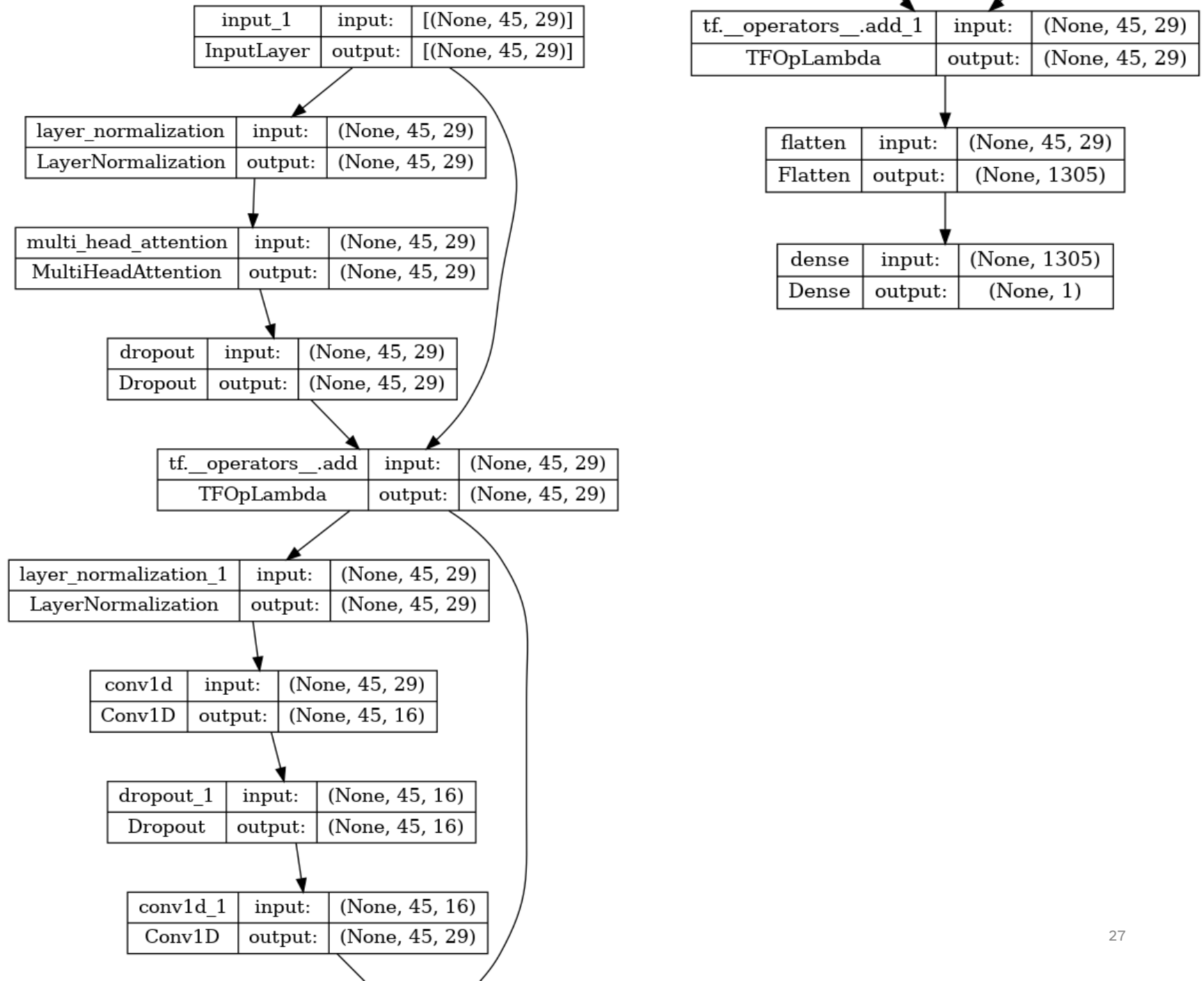
MODEL 1: CNN


input_4 | input: | [(None, 45, 29)]
InputLayer | output: | [(None, 45, 29)]
conv1d_5 | input: | (None, 45, 29)
Conv1D | output: | (None, 45, 16)
dropout_10 | input: | (None, 45, 16)
Dropout | output: | (None, 45, 16)
activation_3 | input: | (None, 45, 16)
Activation | output: | (None, 45, 16)
conv1d_6 | input: | (None, 45, 16)
Conv1D | output: | (None, 45, 16)
dropout_11 | input: | (None, 45, 16)
Dropout | output: | (None, 45, 16)
activation_4 | input: | (None, 45, 16)
Activation | output: | (None, 45, 16)
conv1d_7 | input: | (None, 45, 16)
Conv1D | output: | (None, 45, 16)
dropout_12 | input: | (None, 45, 16)
Dropout | output: | (None, 45, 16)
activation_5 | input: | (None, 45, 16)
Activation | output: | (None, 45, 16)
flatten_3 | input: | (None, 45, 16)
Flatten | output: | (None, 720)
dense_8 | input: | (None, 720)
Dense | output: | (None, 1)

MODEL 2: DNN


input_2 | input: | [(None, 45, 29)]
InputLayer | output: | [(None, 45, 29)]
flatten_1 | input: | (None, 45, 29)
Flatten | output: | (None, 1305)
dense_1 | input: | (None, 1305)
Dense | output: | (None, 2048)
dropout_2 | input: | (None, 2048)
Dropout | output: | (None, 2048)
dense_2 | input: | (None, 2048)
Dense | output: | (None, 512)
dropout_3 | input: | (None, 512)
Dropout | output: | (None, 512)
dense_3 | input: | (None, 512)
Dense | output: | (None, 128)
dropout_4 | input: | (None, 128)
Dropout | output: | (None, 128)
dense_4 | input: | (None, 128)
Dense | output: | (None, 32)
dropout_5 | input: | (None, 32)
Dropout | output: | (None, 32)
dense_5 | input: | (None, 32)
Dense | output: | (None, 8)
dropout_6 | input: | (None, 8)
Dropout | output: | (None, 8)
dense_6 | input: | (None, 8)
Dense | output: | (None, 1)

MODEL 3: TRANSFORMER



2. GRID-SEARCH (CNN) – MEAN VALUES

param_act	param_do_batch_norm	param_filters	param_kernel	param_num_layers	harmonic_mean_acc_rec	Accuracy	Recall
gelu	FALSE	16	2	3	0.873798	84.082 ± 0.53%	89.456 ± 0.314%
gelu	FALSE	64	1	3	0.868311	84.155 ± 0.808%	88.496 ± 0.981%
gelu	FALSE	16	1	3	0.864541	81.377 ± 0.788%	89.768 ± 1.696%
gelu	FALSE	16	2	6	0.86378	83.208 ± 1.127%	88.37 ± 0.823%
gelu	FALSE	64	2	6	0.788001	86.652 ± 8.541%	74.79 ± 10.704%

- All values above represent the mean values obtained when training using each model

2. GRID-SEARCH (DNN) – MEAN VALUES

param_act	param_dims	param_dropout	harmonic_mean_acc_rec	Accuracy	Recall
gelu	[2048, 512, 128, 32, 8]	0	0.901	87.773 ± 0.803%	91.603 ± 6.633%
elu	[32, 32, 32, 32]	0.4	0.9	85.64 ± 0.758%	92.822 ± 6.464%
relu	[32, 32, 32, 32]	0	0.9	86.147 ± 0.739%	92.412 ± 8.164%
elu	[32, 32, 32, 32]	0	0.898	86.802 ± 0.842%	91.603 ± 6.633%
gelu	[32, 32, 32, 32]	0.2	0.897	85.482 ± 0.737%	92.372 ± 7.095%

- All values above represent the mean values obtained when training using each model

2. GRID-SEARCH (TRANSFORMER)– MEAN VALUES

param_ff_dim	param_kernel_size	param_num_transformer_blocks	param_dropout	param_head_size	param_num_heads	harmonic_mean_acc_rec	Accuracy	Recall
16	3	1	0.2	256	3	0.891	86.37 ± 1.116%	90.718 ± 2.441%
32	5	5	0	128	3	0.89	85.896 ± 1.165%	90.883 ± 1.199%
4	3	1	0.5	512	3	0.889	80.904 ± 6.021%	94.365 ± 4.851%
4	5	1	0.5	128	5	0.888	85.012 ± 3.329%	91.139 ± 2.968%
16	1	1	0	512	3	0.886	86.286 ± 4.085%	89.949 ± 6.178%

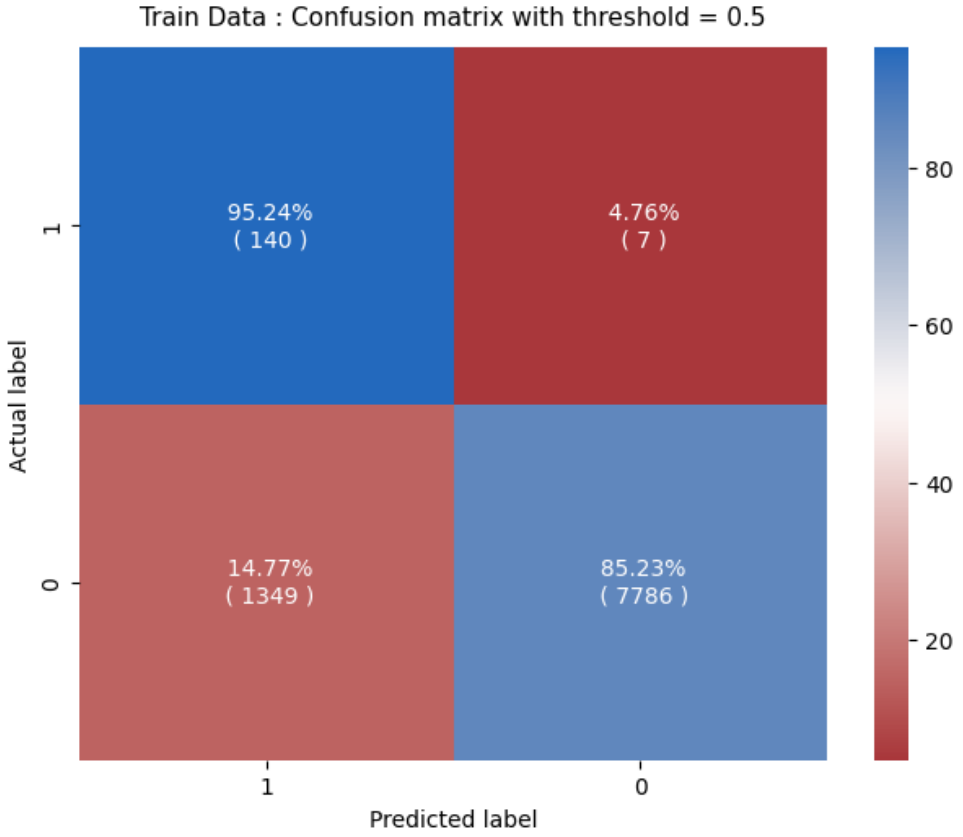
- All values above represent the mean values obtained when training using each model

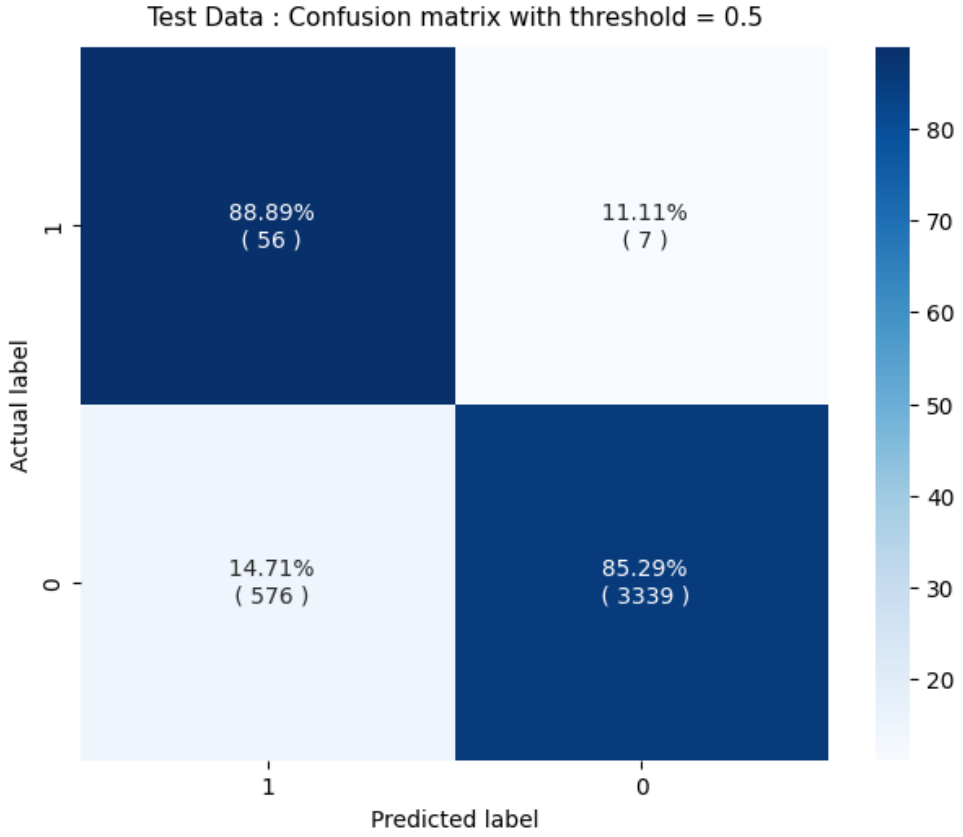
2. GRID-SEARCH – BEST VALUES FOUND

Model	Accuracy ± std	Recall ± std
CNN	0.84 ± 0.005	0.89 ± 0.03
DNN	0.88 ± 0.08	0.91 ± 0.06
Transformer	0.86 ± 0.01	0.91 ± 0.02

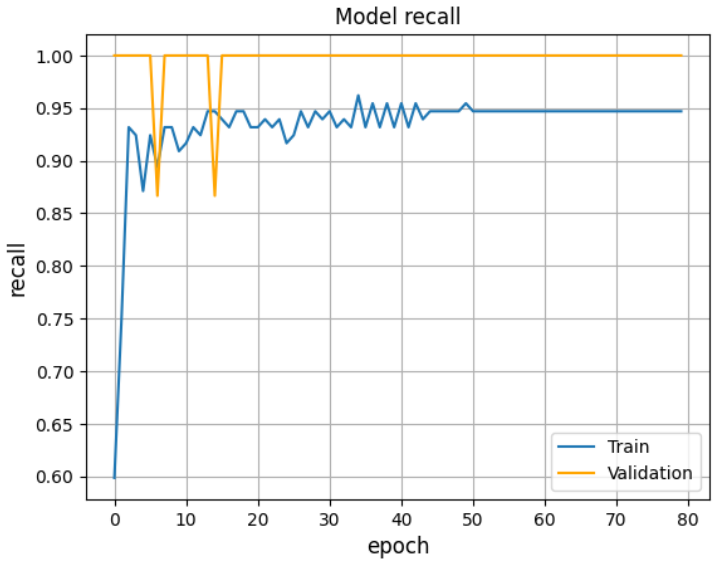
- Overall, comparing all these three models, Transformer has the better performance and CNN shows the least desirable performance.

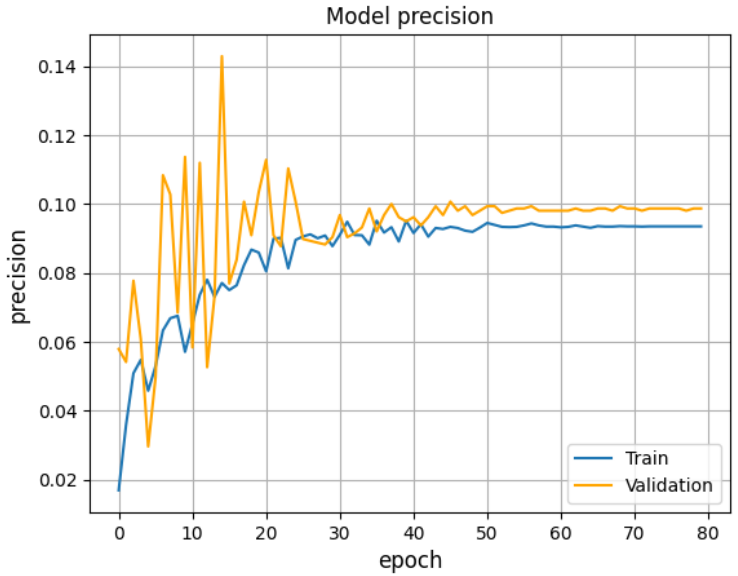
2. GRID-SEARCH COMPARISON: TRAINING ON BEST CNN MODEL

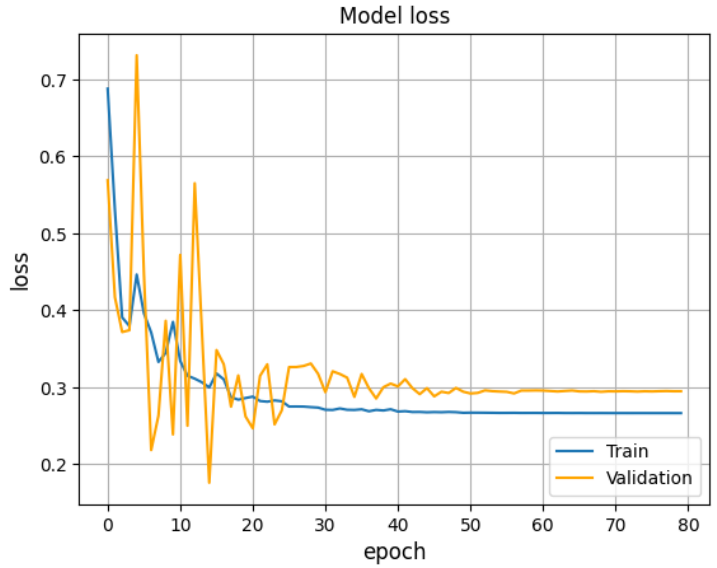


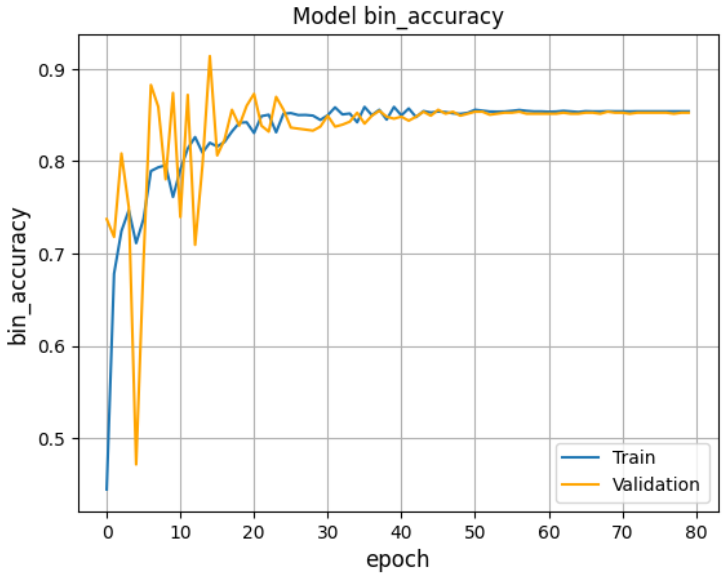


2. GRID-SEARCH COMPARISON: TRAINING ON BEST CNN MODEL

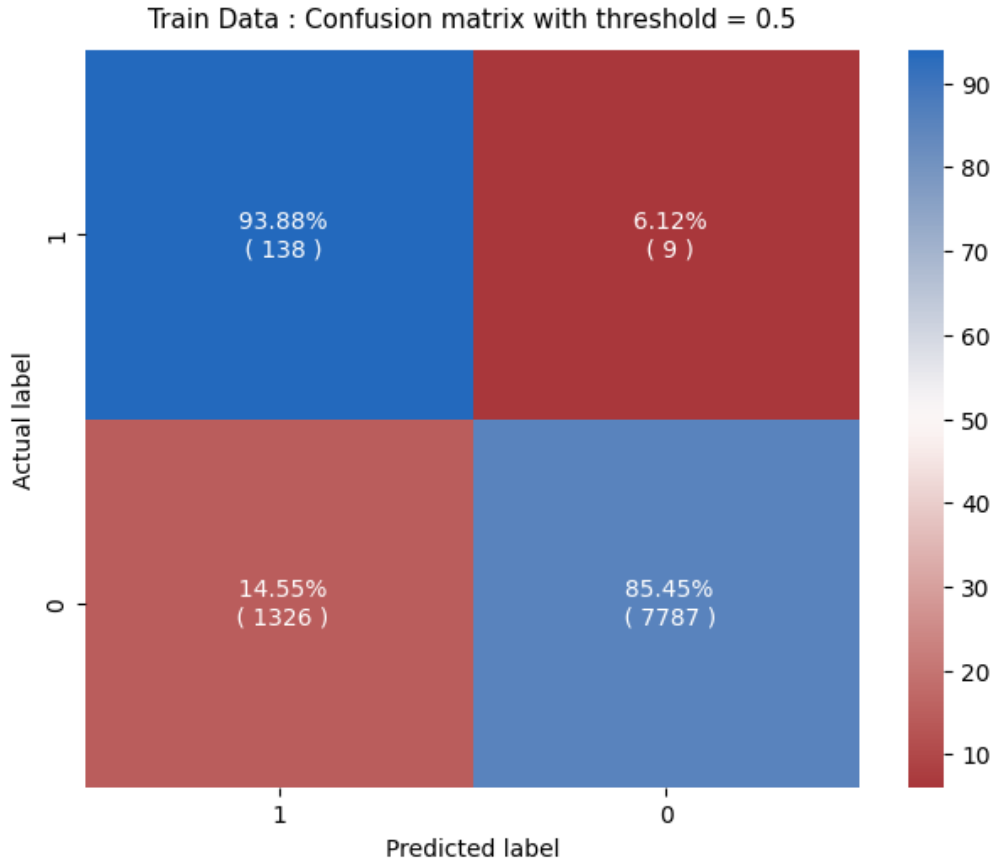


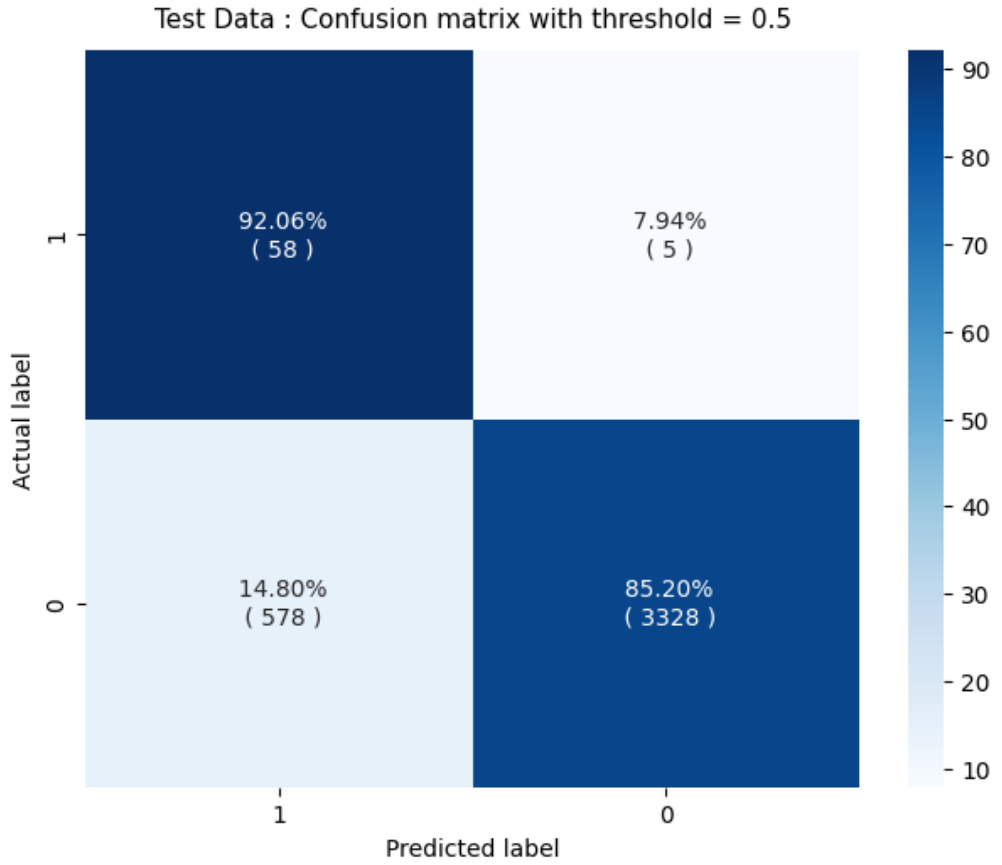




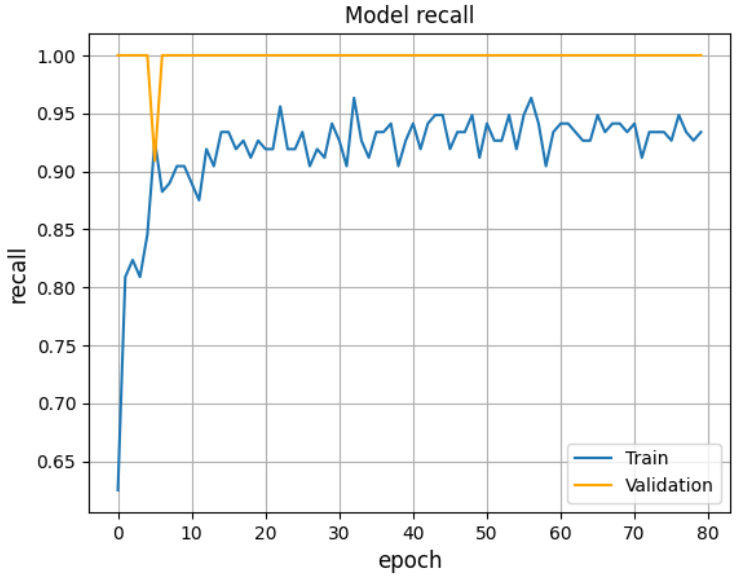


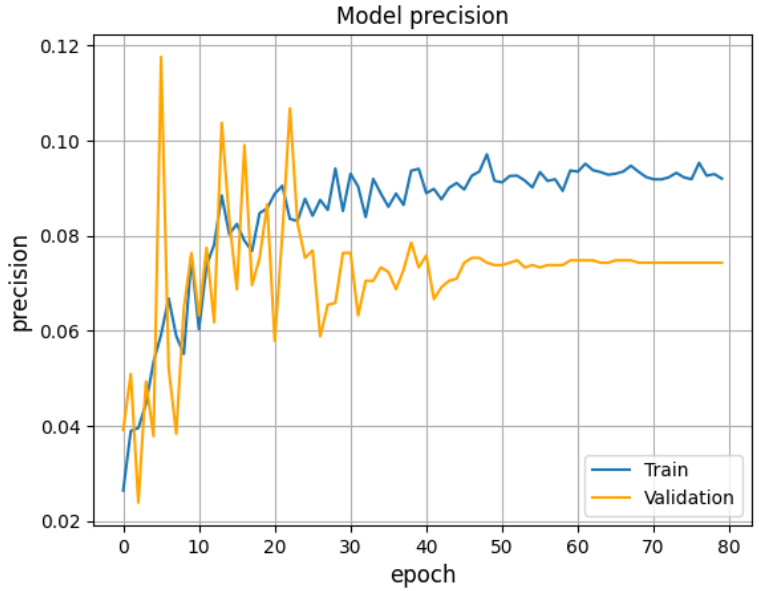
2. GRID-SEARCH COMPARISON: TRAINING ON BEST DNN MODEL

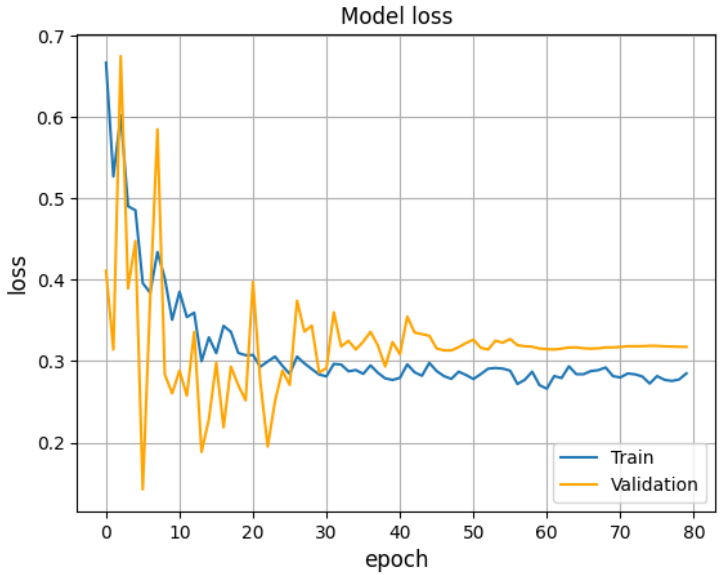


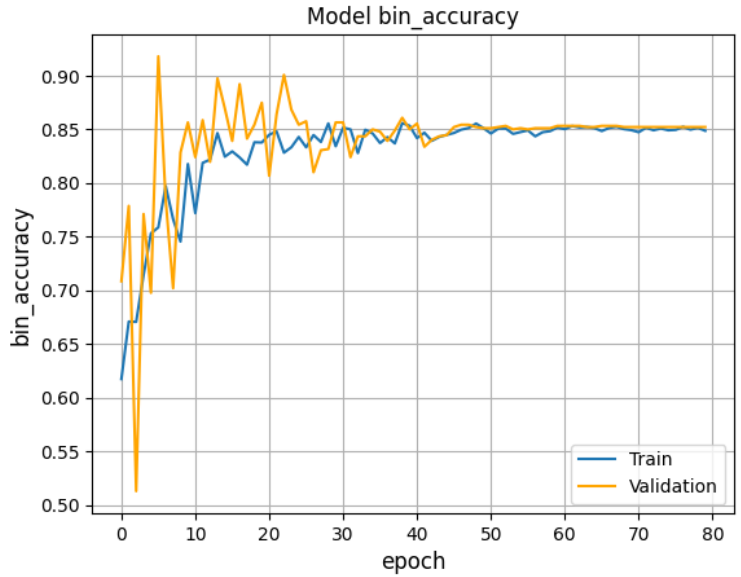


2. GRID-SEARCH COMPARISON: TRAINING ON BEST DNN MODEL

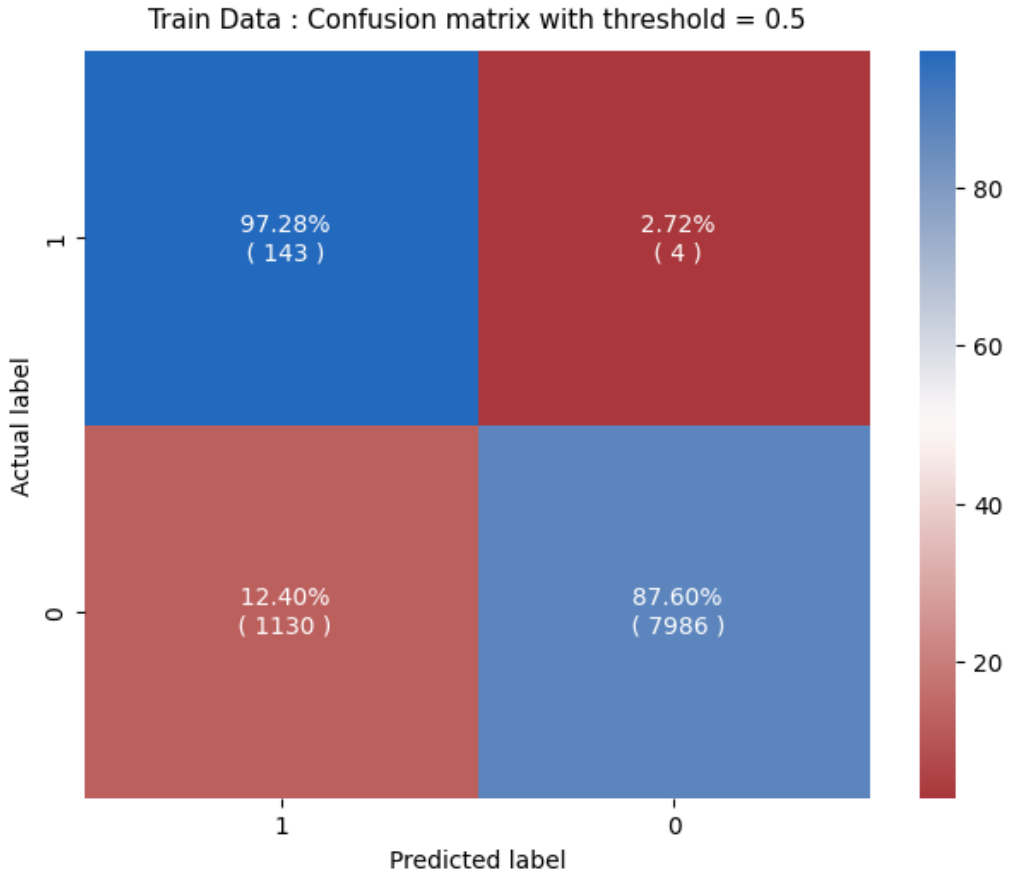


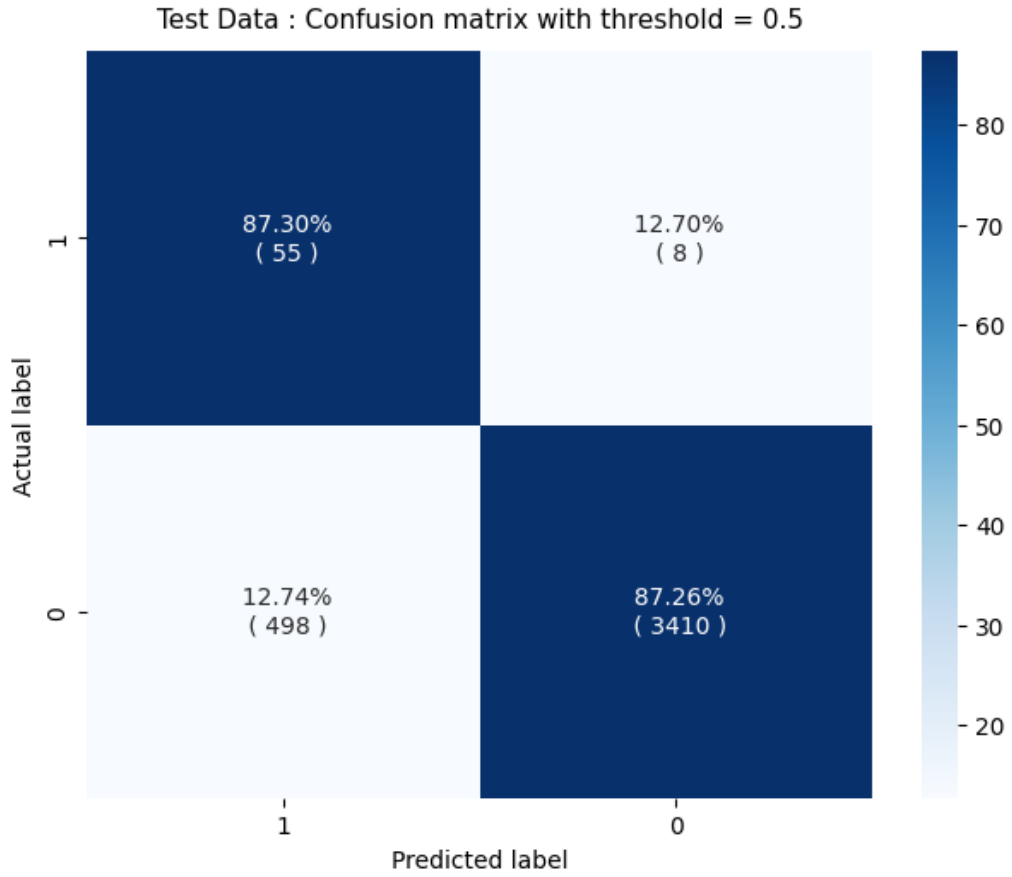




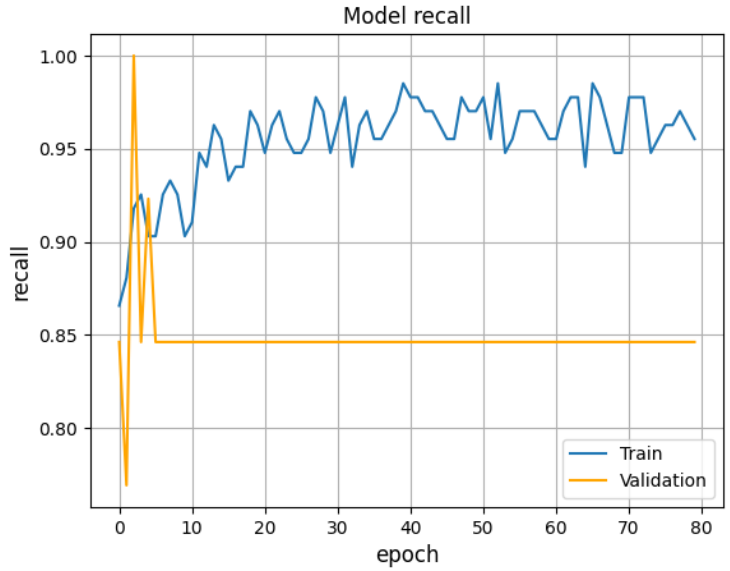


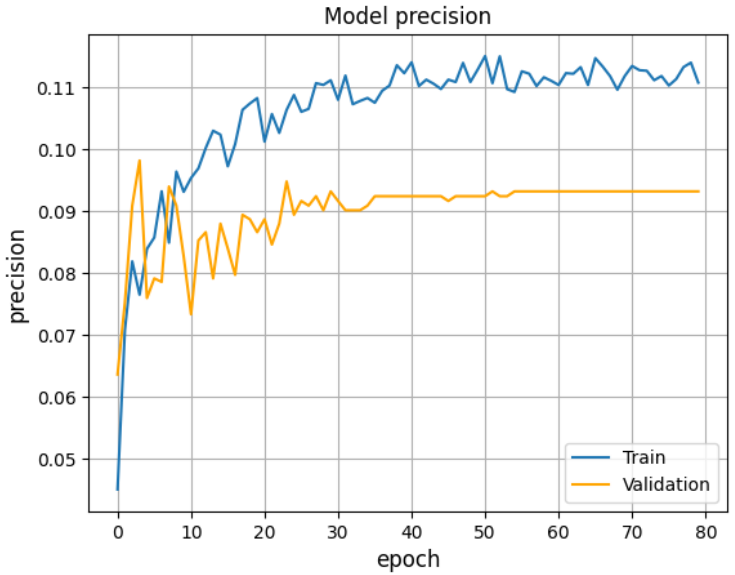
2. GRID-SEARCH COMPARISON: TRAINING ON BEST TRANSFORMER MODEL

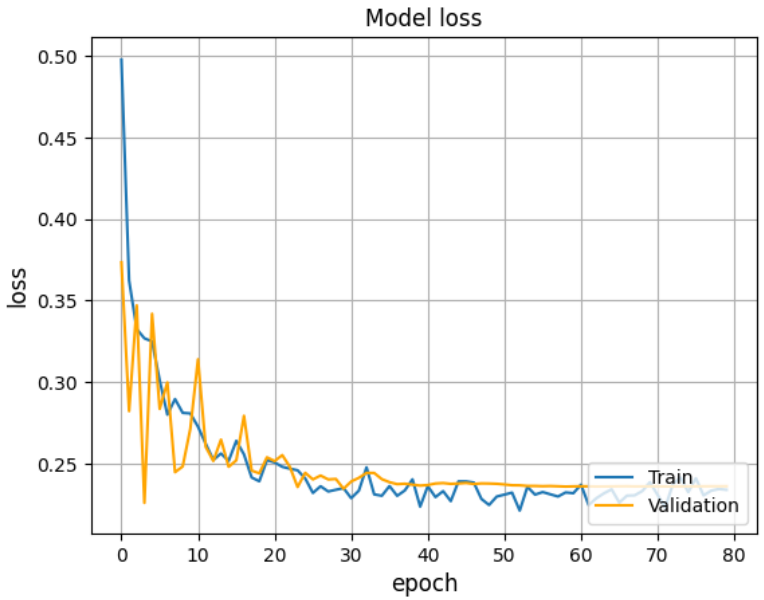


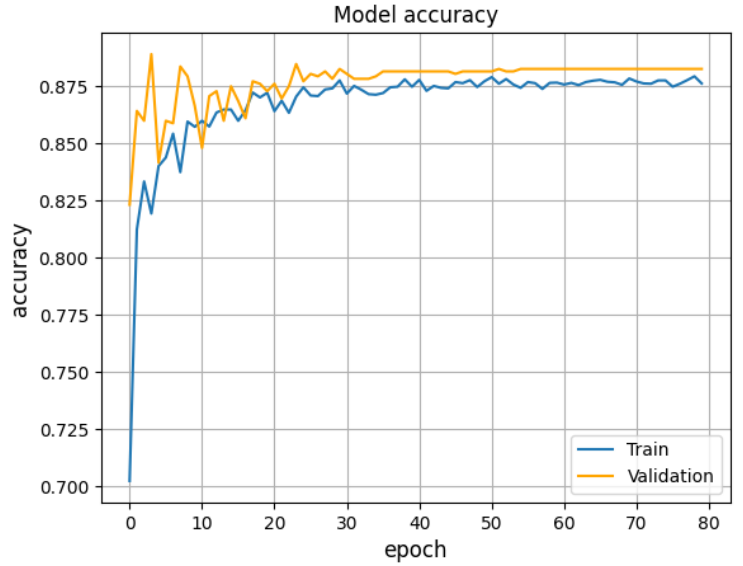


2. GRID-SEARCH COMPARISON: TRAINING ON BEST TRANSFORMER MODEL









3. WHY TRANSFORMERS

MEAN RESULTS

- Better Accuracy
- Better Recall
- Better Precision

STANDARD DEVIATION

- Transformers had comparable values to CNN, signifying the reliability of the model

IV.
IMPROVEMENT OF BEST MODEL

1. Grid-Search on parameters
2. Heatmap Visualization on Results
3. Behaviour across long times
4. Threshold Moving
5. Final Results

1. GRID-SEARCH ON PARAMETERS

OBJECTIVE

Select optimal set of data parameters:

- Window Length (L)
- Prediction Length (dt)
- Prediction Window (WR)
- Applying PCA

VALUES USED

- **L:** [15, 30, 45, 60, 75, 90]
- **dt:** [10, 25, 40, 55, 70, 85]
- **WR:** [0, 1]
- **PCA:** [False, True]

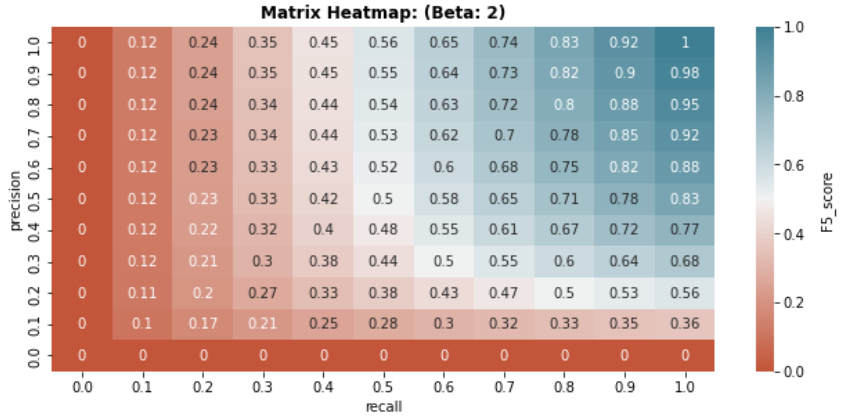
OPTIMAL SET

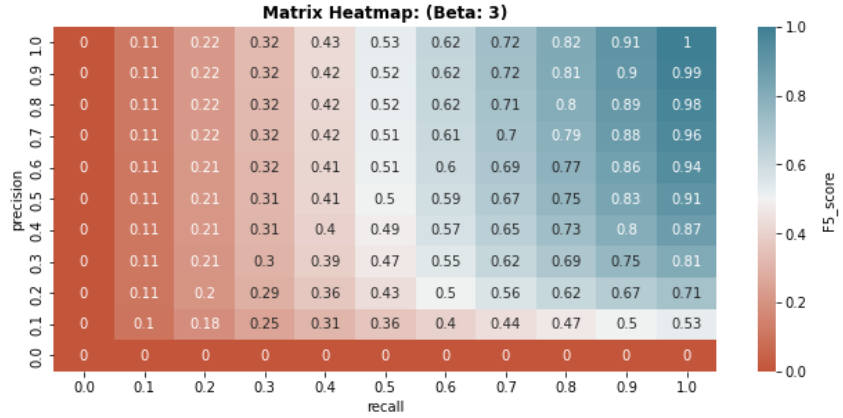
- **L:** 45
- **dt:** 10
- **W:** 1
- **PCA:** True

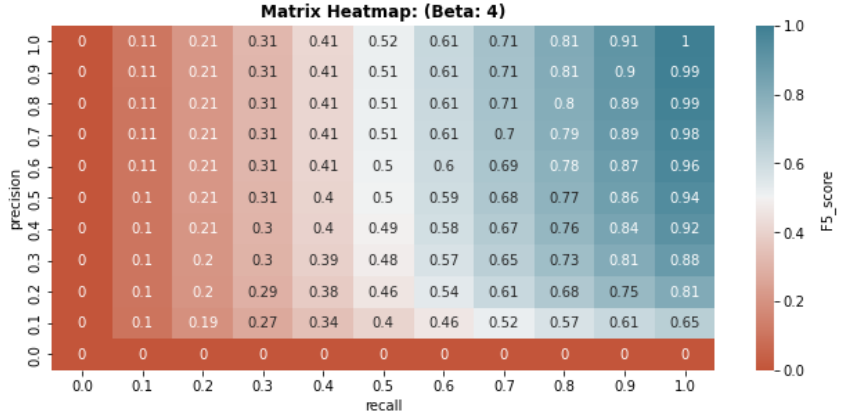
2. HEATMAP VISUALIZATION OF RESULTS – THE F-BETA SCORE

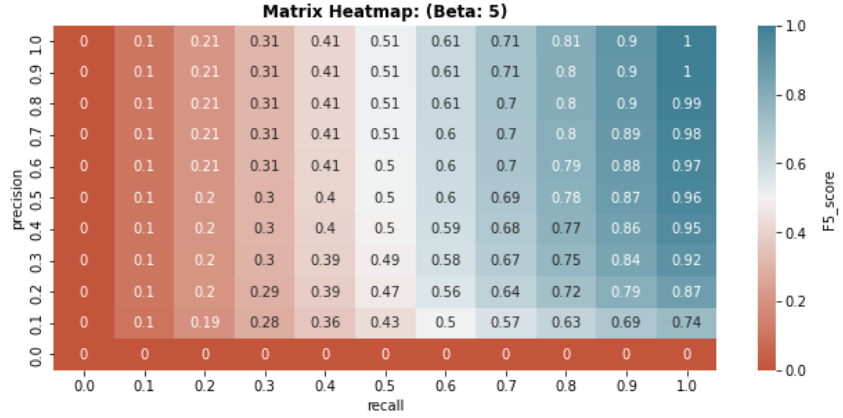
- Weighted Harmonic Mean between Precision and Recall, with a weight factor (beta) that assigns importance to recall

- Notice how the plot starts out as horizontal for low beta and slowly starts leaning vertical as beta increases. This signals recall's increasing impact on the metric.

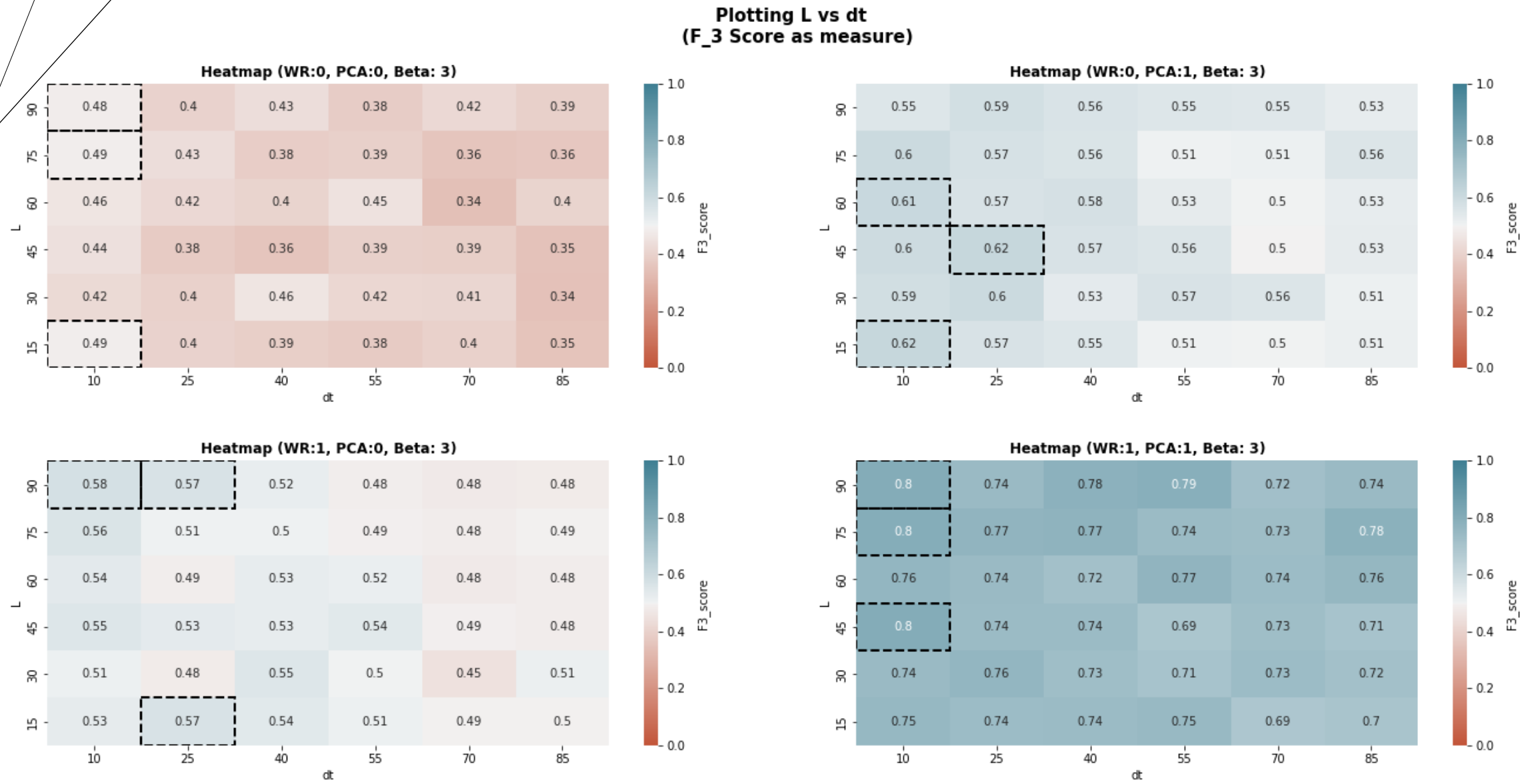




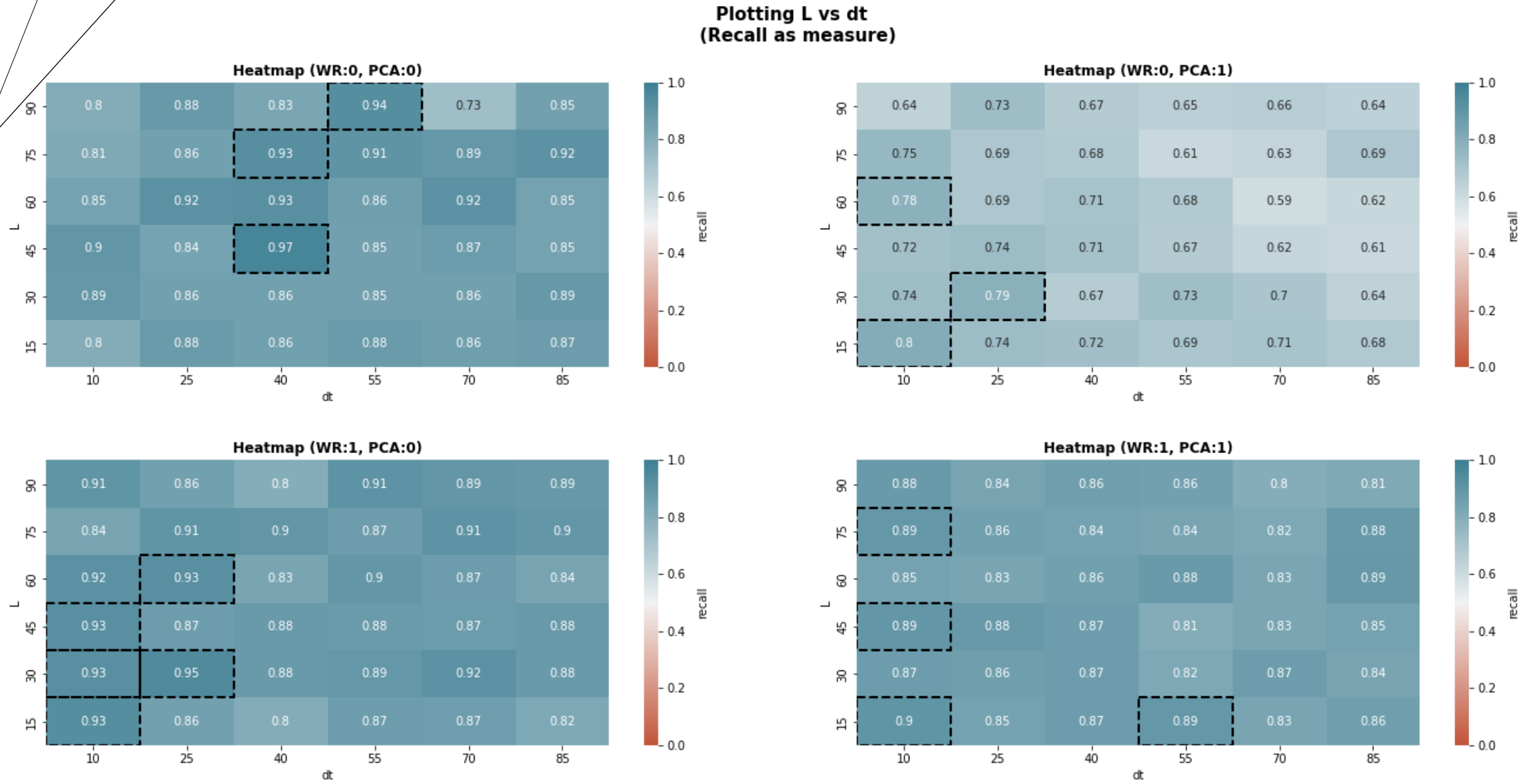




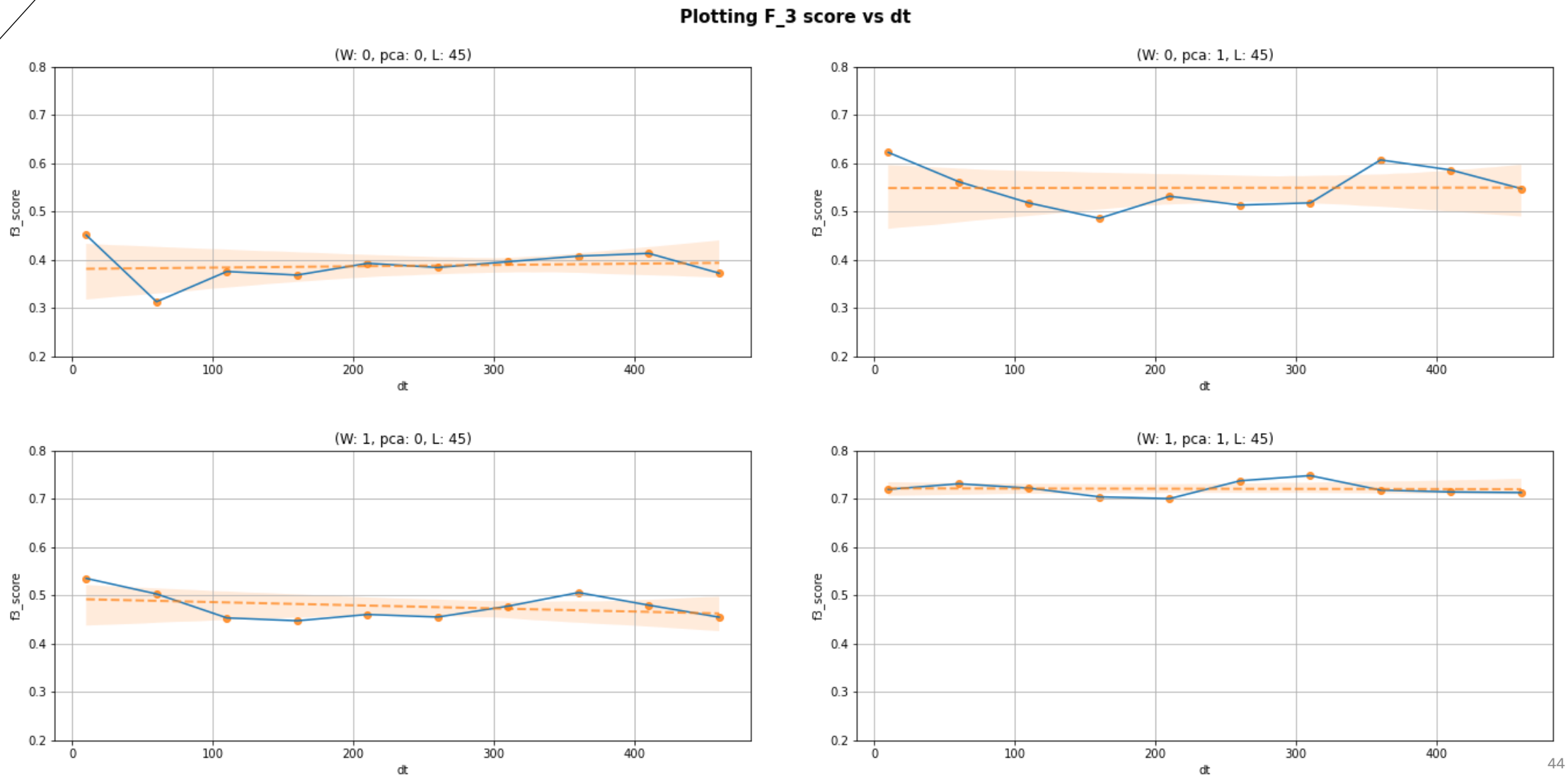
2. HEATMAP VISUALIZATION OF RESULTS



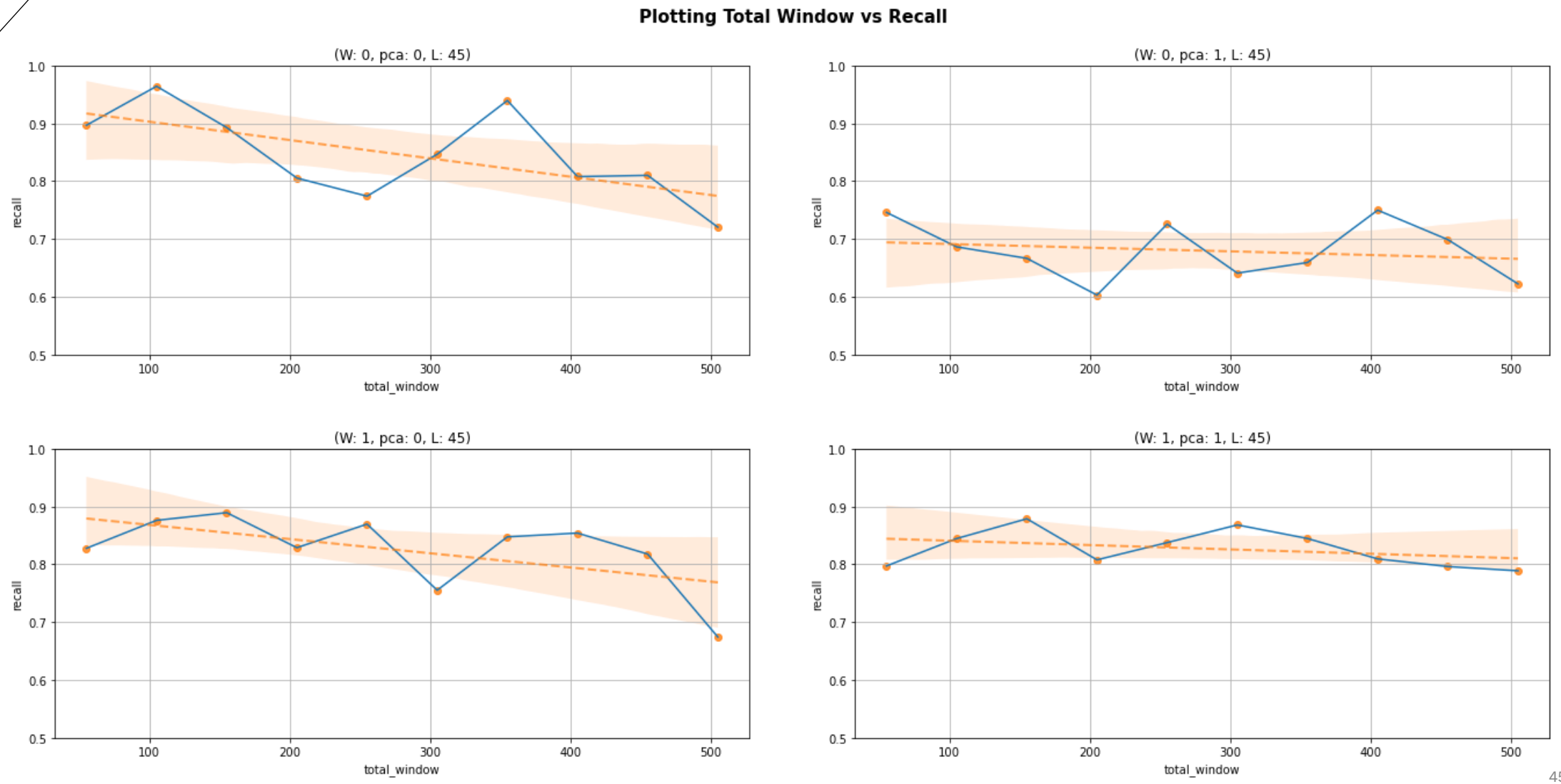
2. HEATMAP VISUALIZATION OF RESULTS



3. BEHAVIOUR ACROSS LONG TIMES



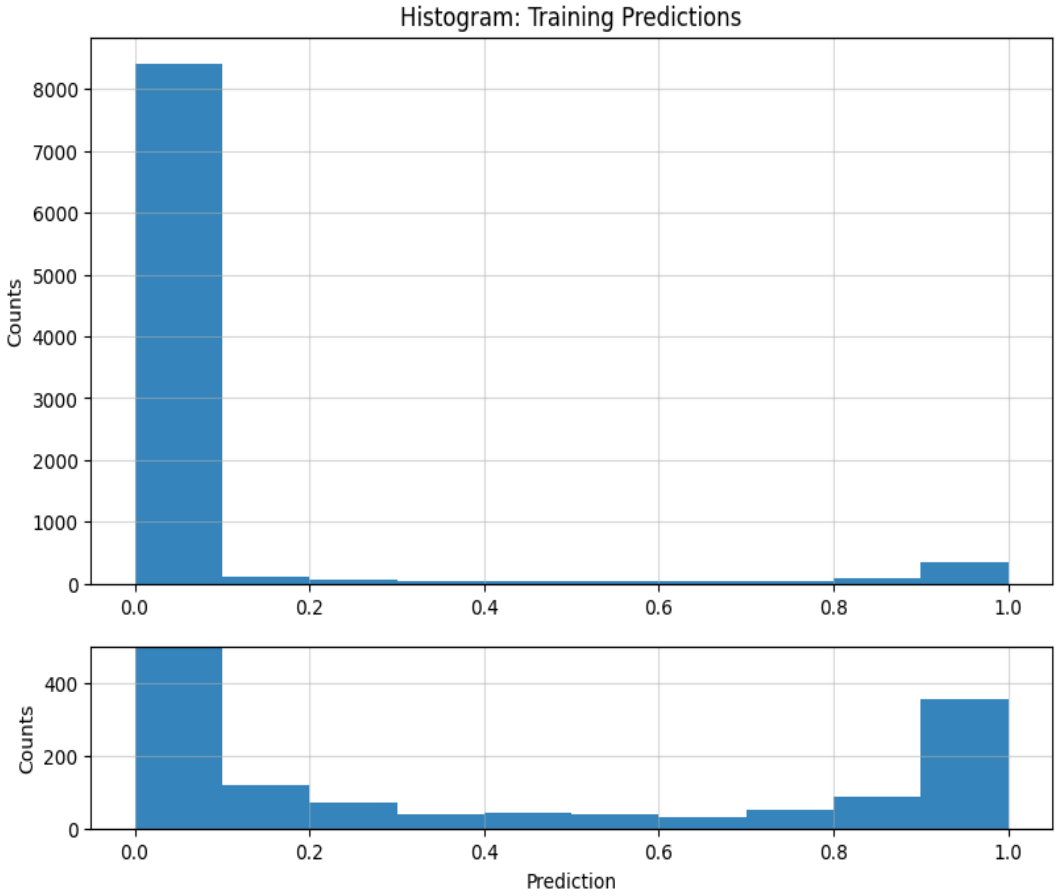
3. BEHAVIOUR ACROSS LONG TIMES



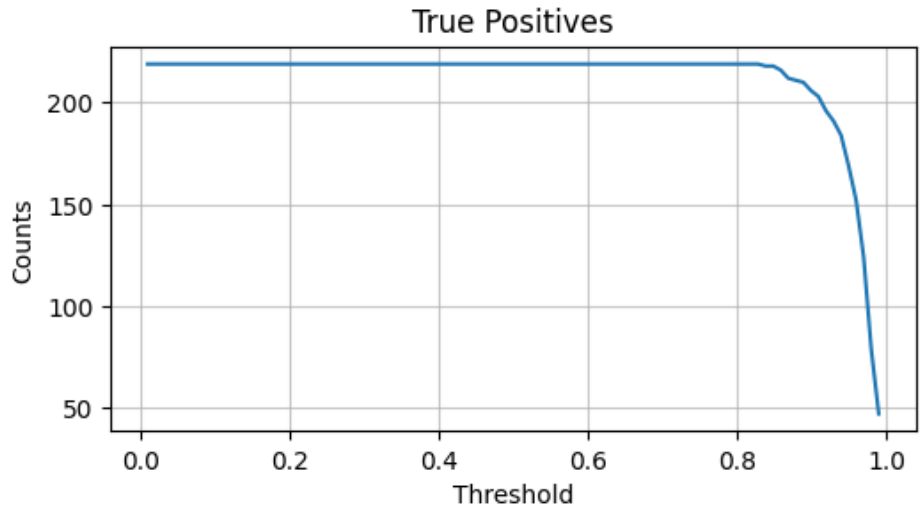
4. THRESHOLD MOVING

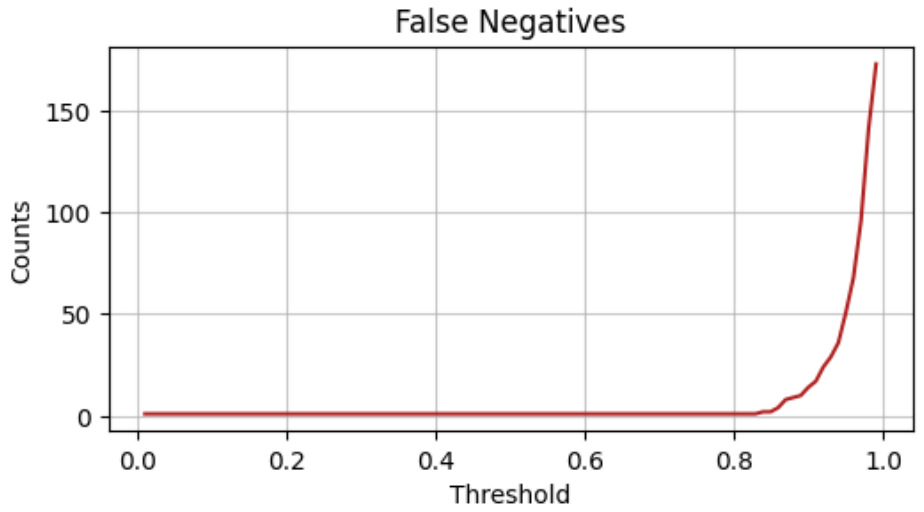
WHY MOVE THE THRESHOLD?

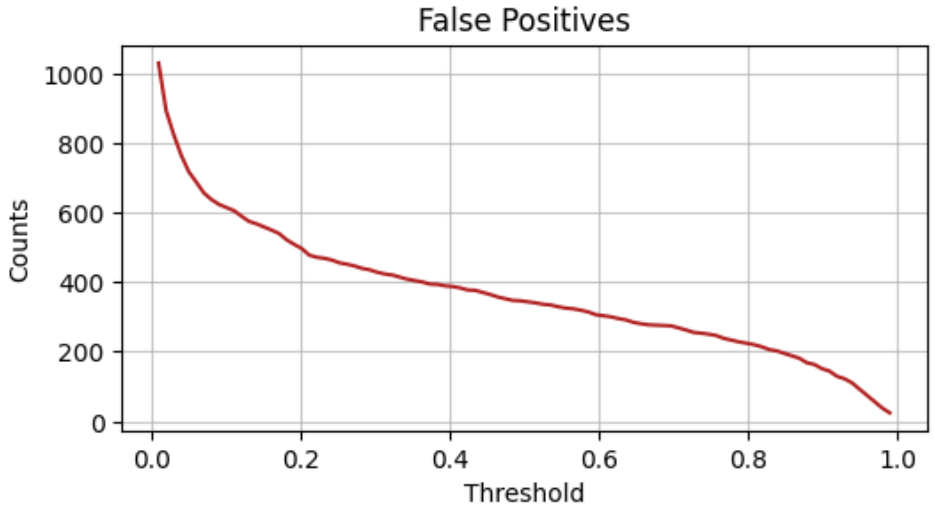
- Usually, in binary classification problems, the threshold is set to 0.5
- But with highly imbalanced data here, this might not be optimal. Moving the threshold helps us find a better balance between correctly identifying the minority class (increasing recall) while keeping an acceptable amount of false positives (precision).
- Lowering the threshold makes the model more sensitive to the minority class (ones): (High recall, low precision)
- Increasing the threshold makes the model behave inversely, preferring precision over recall.
- Moving the threshold to a lower or upper extreme can incentivize the model to just predict ones or zeros all the time. Thus, we need to take into account this too.

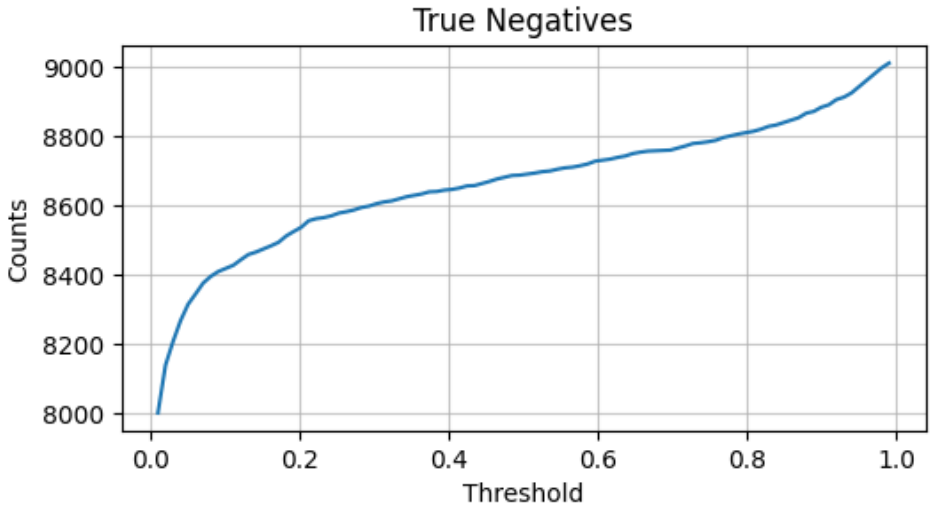


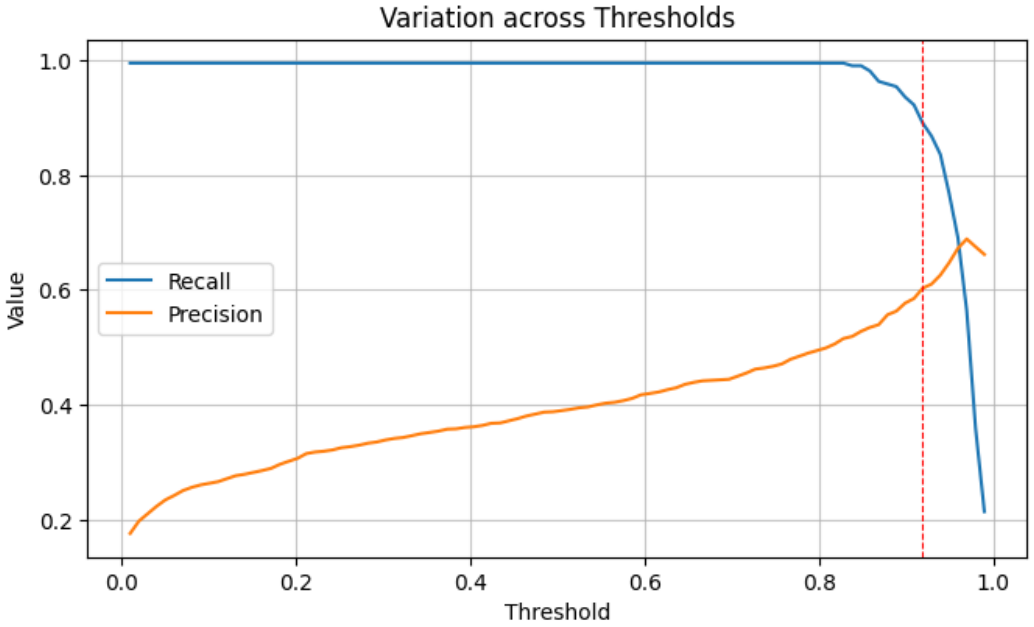
4. THRESHOLD MOVING – VARIATION

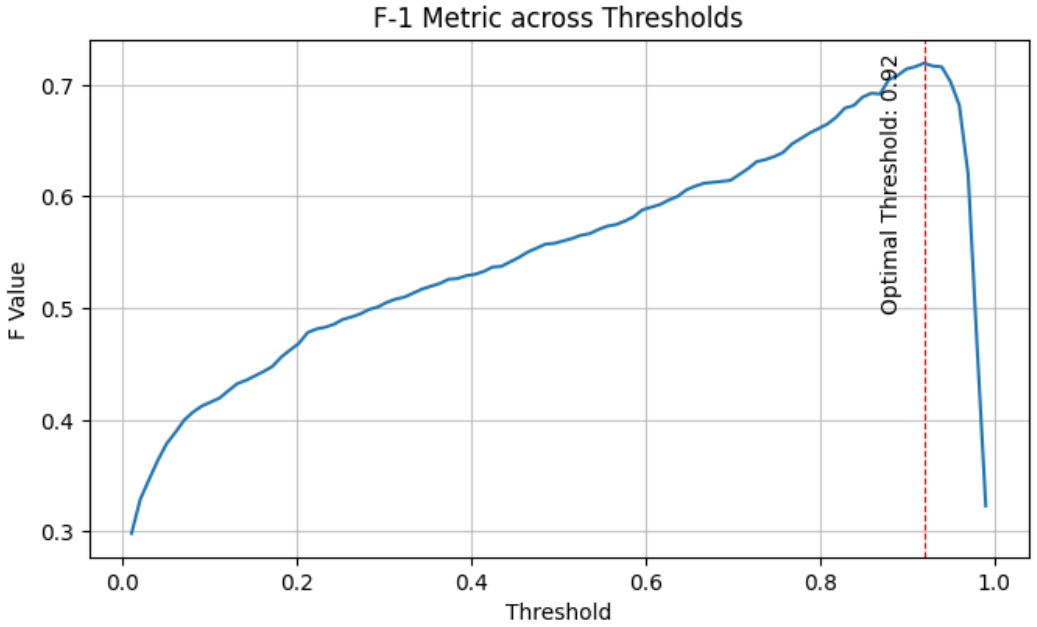


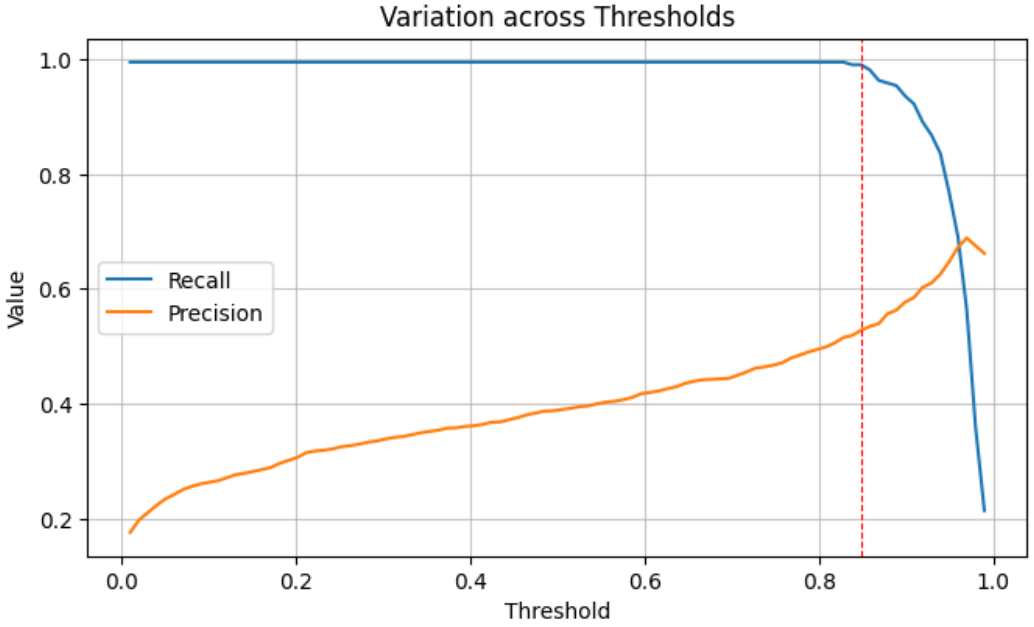


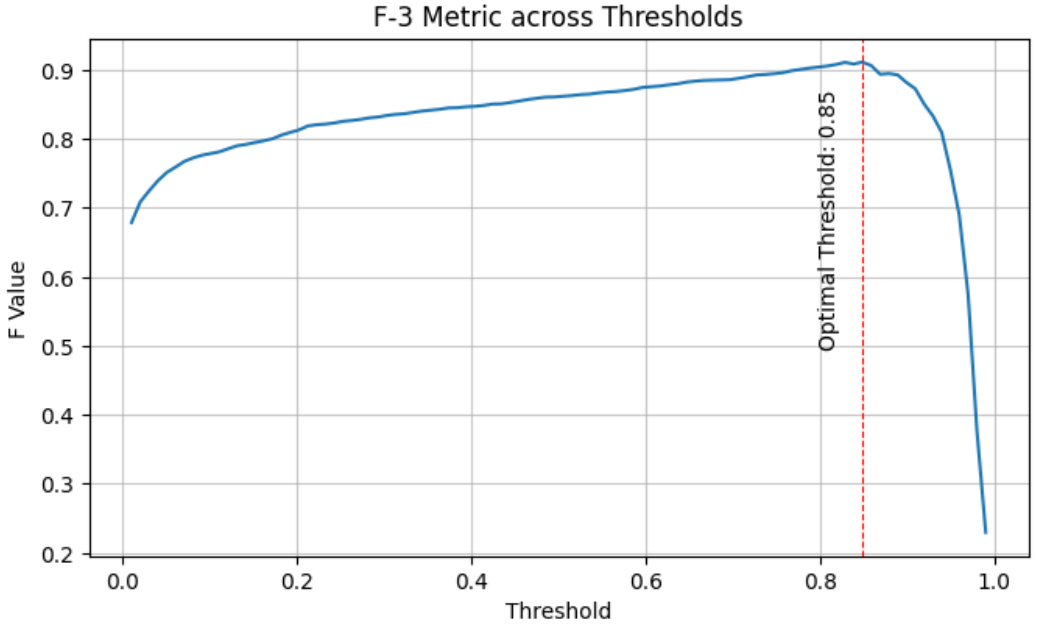





Variation across Thresholds
Recall
Precision
Value
Threshold
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0

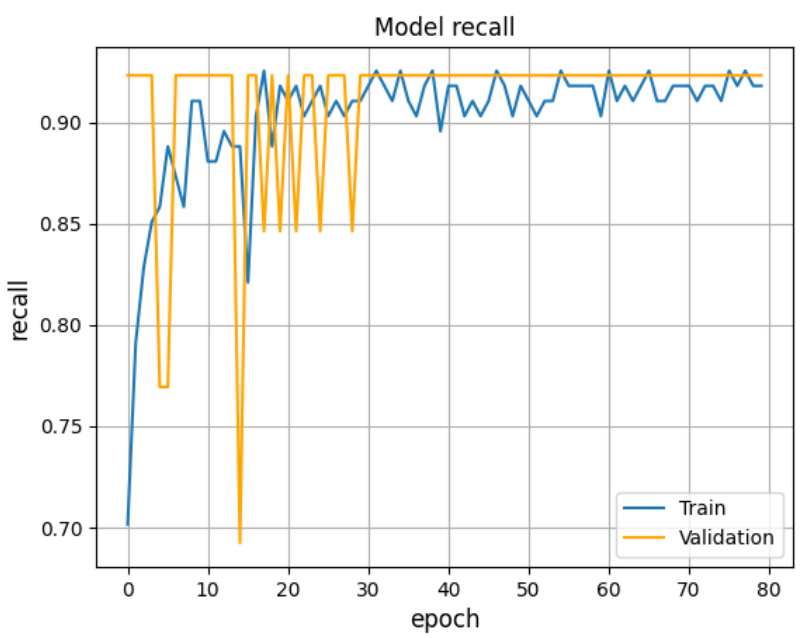

F-1 Metric across Thresholds
Optimal Threshold: 0.92
F Value
Threshold
0.7
0.6
0.5
0.4
0.3
0.0
0.2
0.4
0.6
0.8
1.0

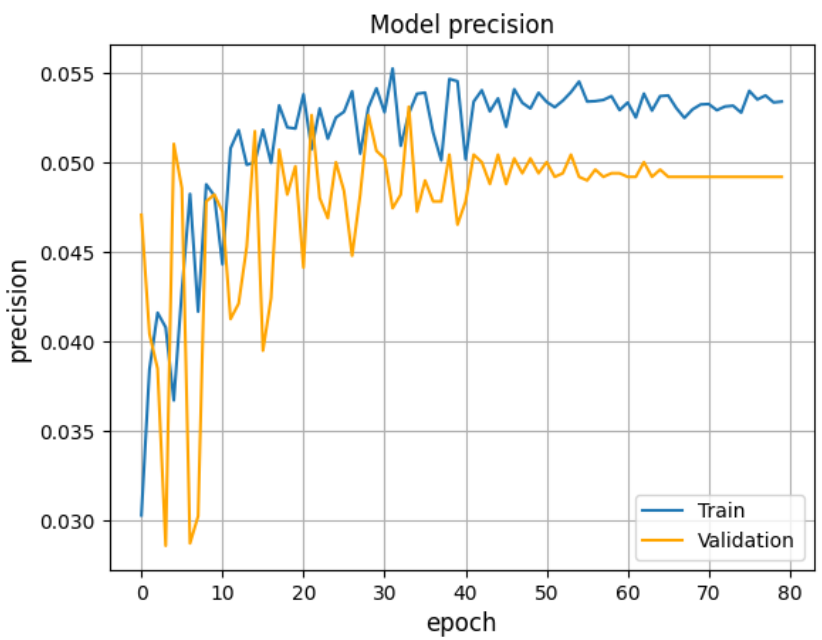

Variation across Thresholds
Recall
Precision
Value
Threshold
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0

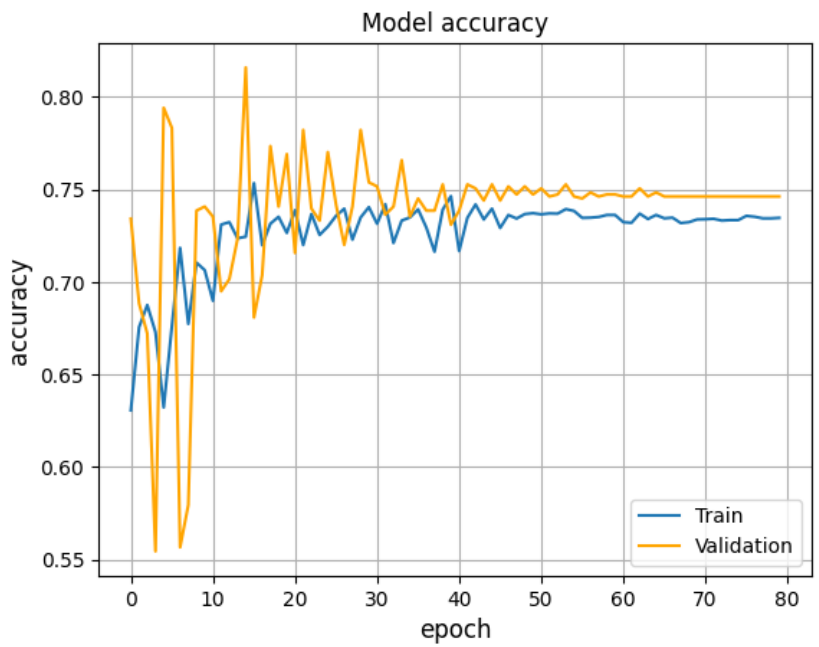

F-3 Metric across Thresholds
Optimal Threshold: 0.85
F Value
Threshold
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.0
0.2
0.4
0.6
0.8
1.0

5. IMPROVED RESULTS OF BEST MODEL (BEFORE IMPROVEMENT)

DATA PARAMETERS: (WR:0, PCA:FALSE, L:45, DT: 10)



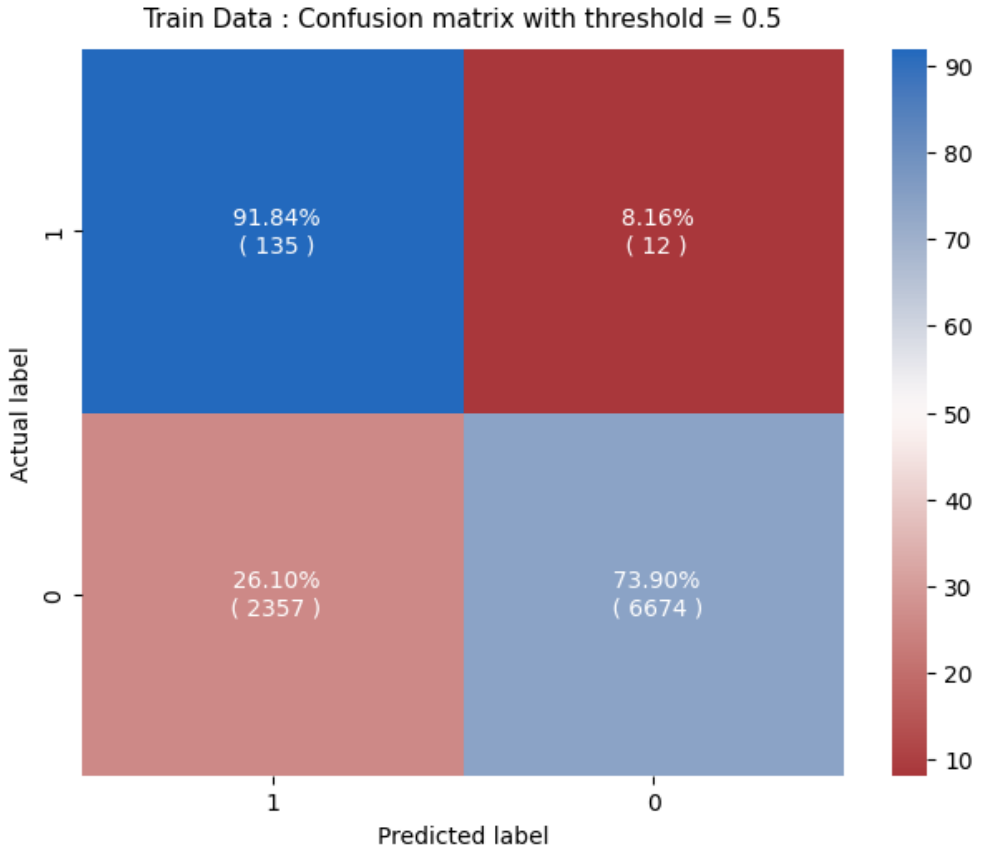


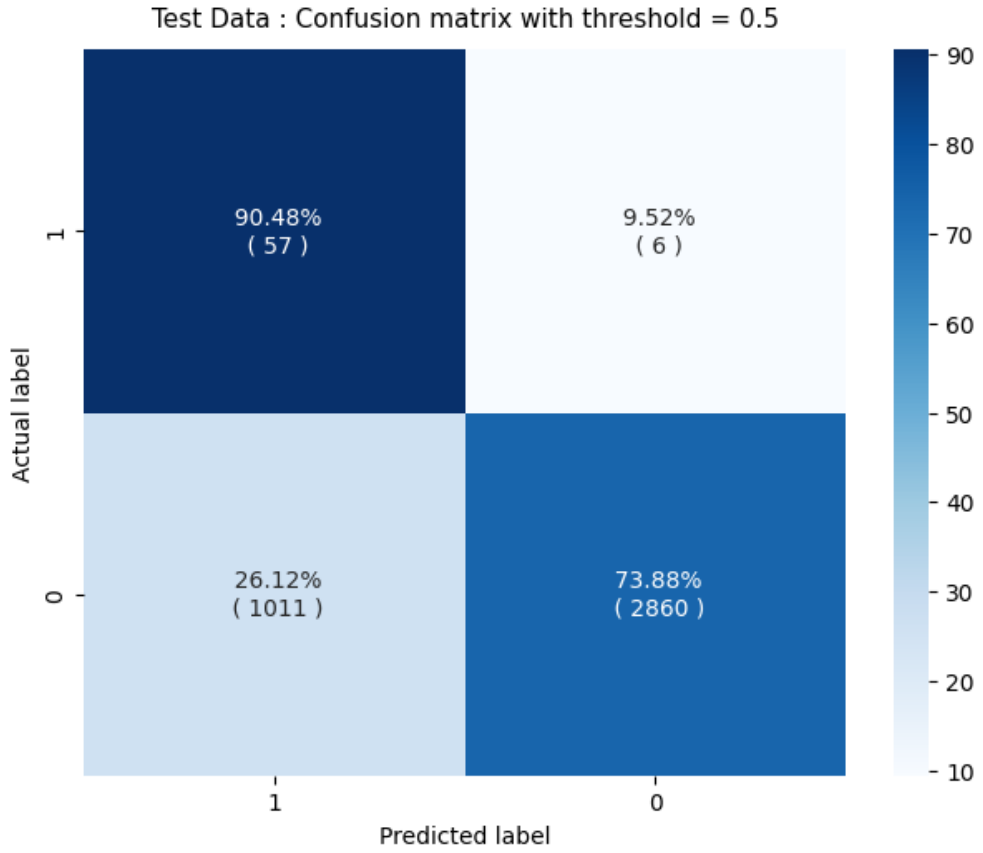


5. IMPROVED RESULTS OF BEST MODEL (BEFORE IMPROVEMENT)

DATA PARAMETERS: (WR:0, PCA:FALSE, L:45, DT: 10)

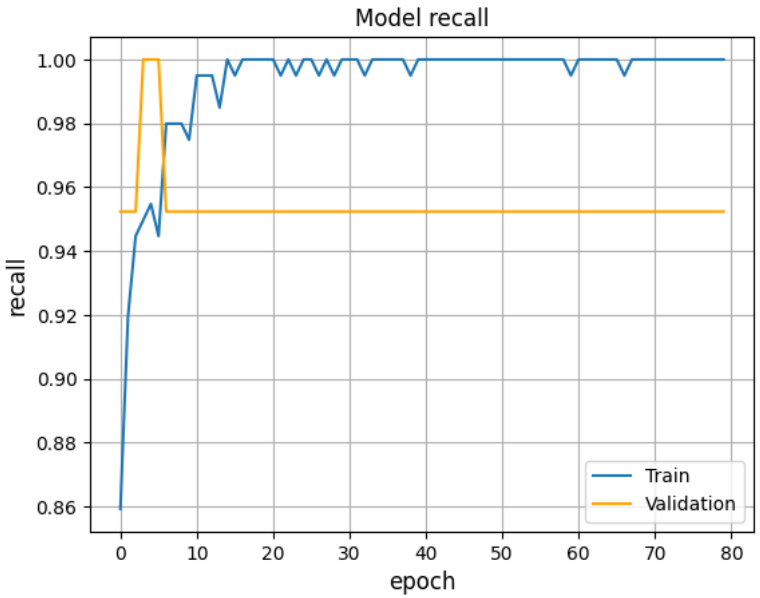
THRESHOLD: 0.5

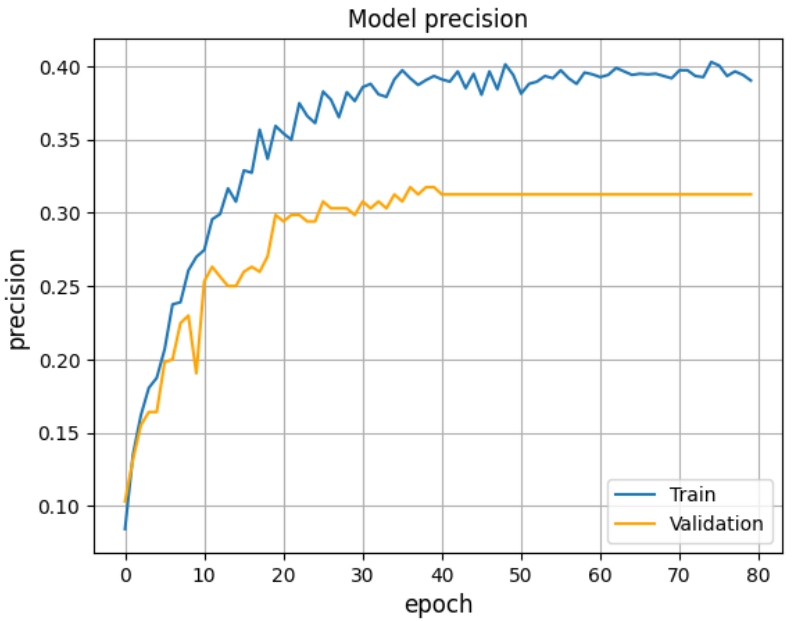


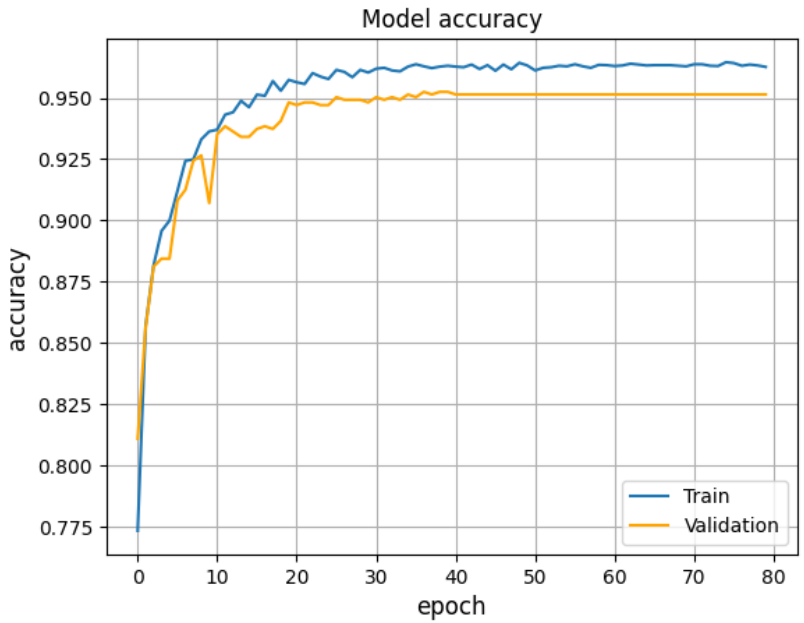


5. IMPROVED RESULTS OF BEST MODEL (AFTER IMPROVEMENT)

DATA PARAMETERS: (WR:1, PCA:TRUE, L:45, DT: 10)



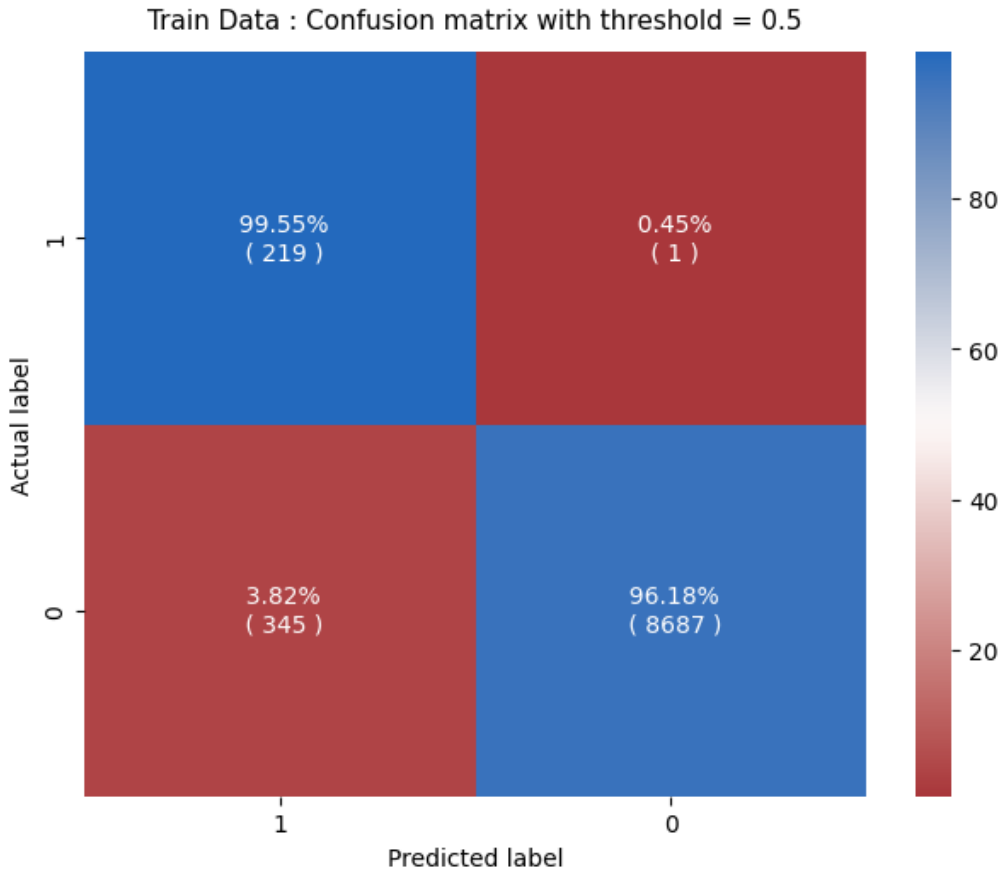


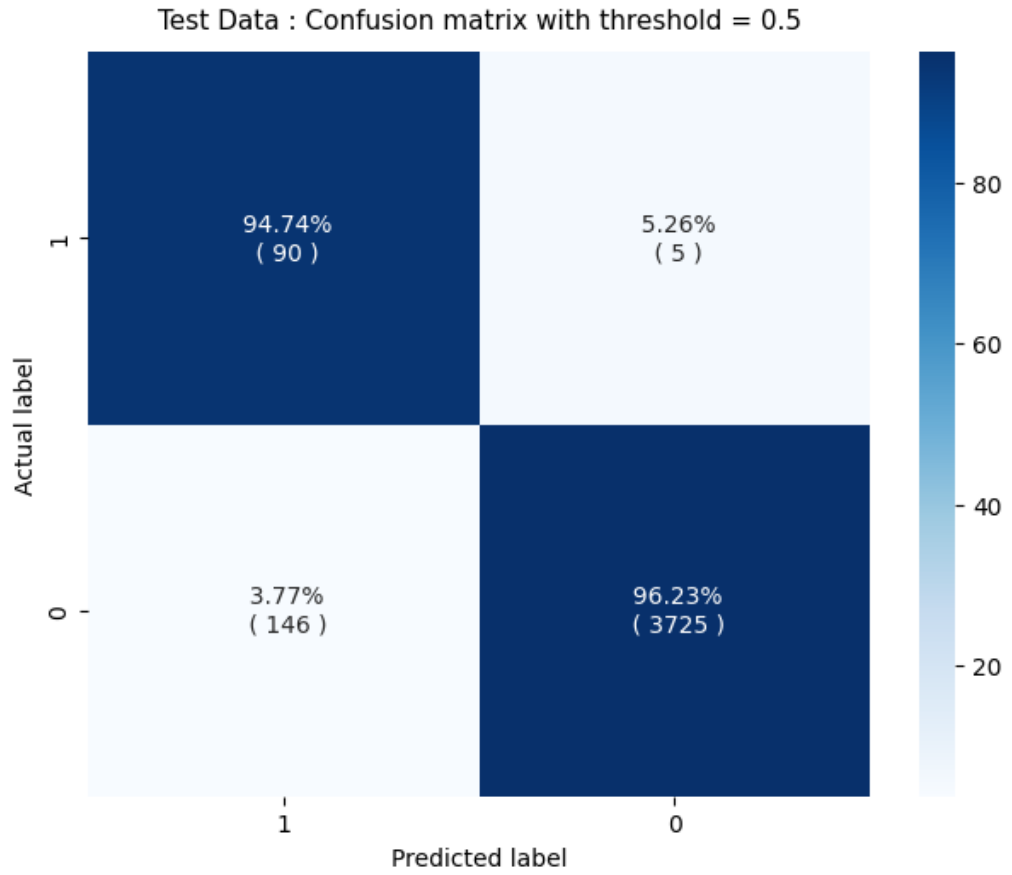


5. IMPROVED RESULTS OF BEST MODEL (AFTER IMPROVEMENT)

DATA PARAMETERS: (WR:1, PCA:TRUE, L:45, DT: 10)

THRESHOLD: 0.5

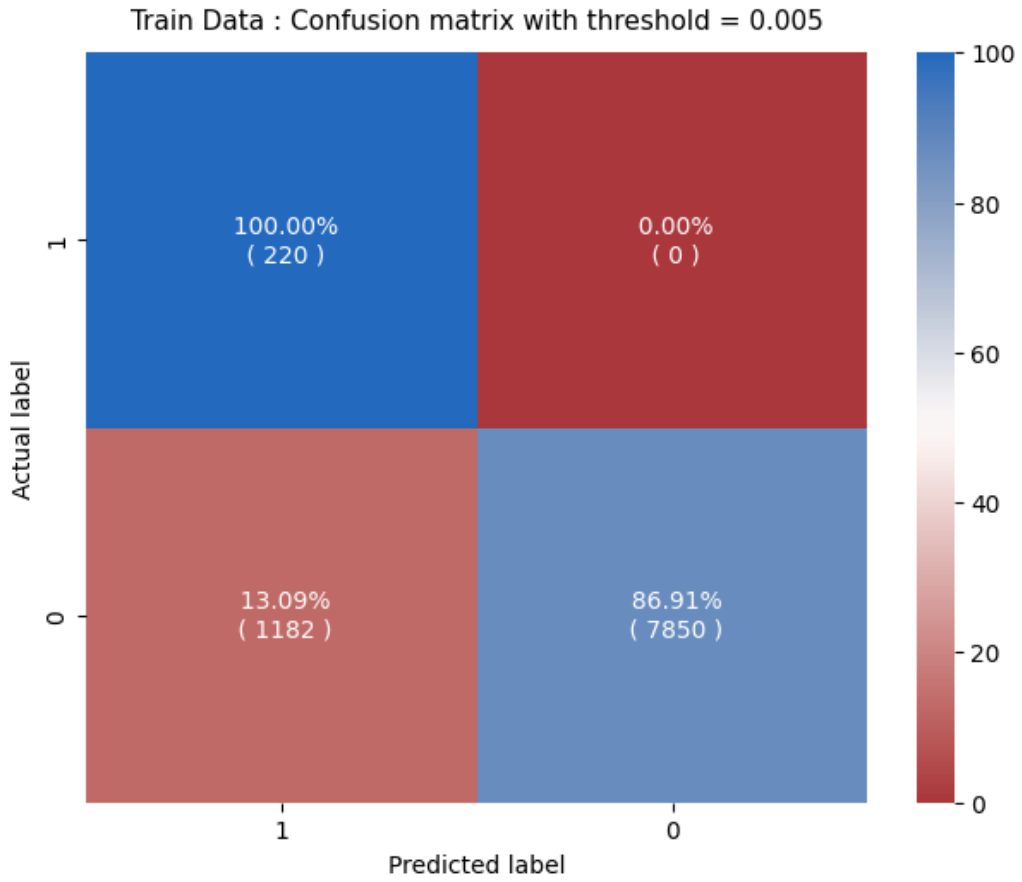


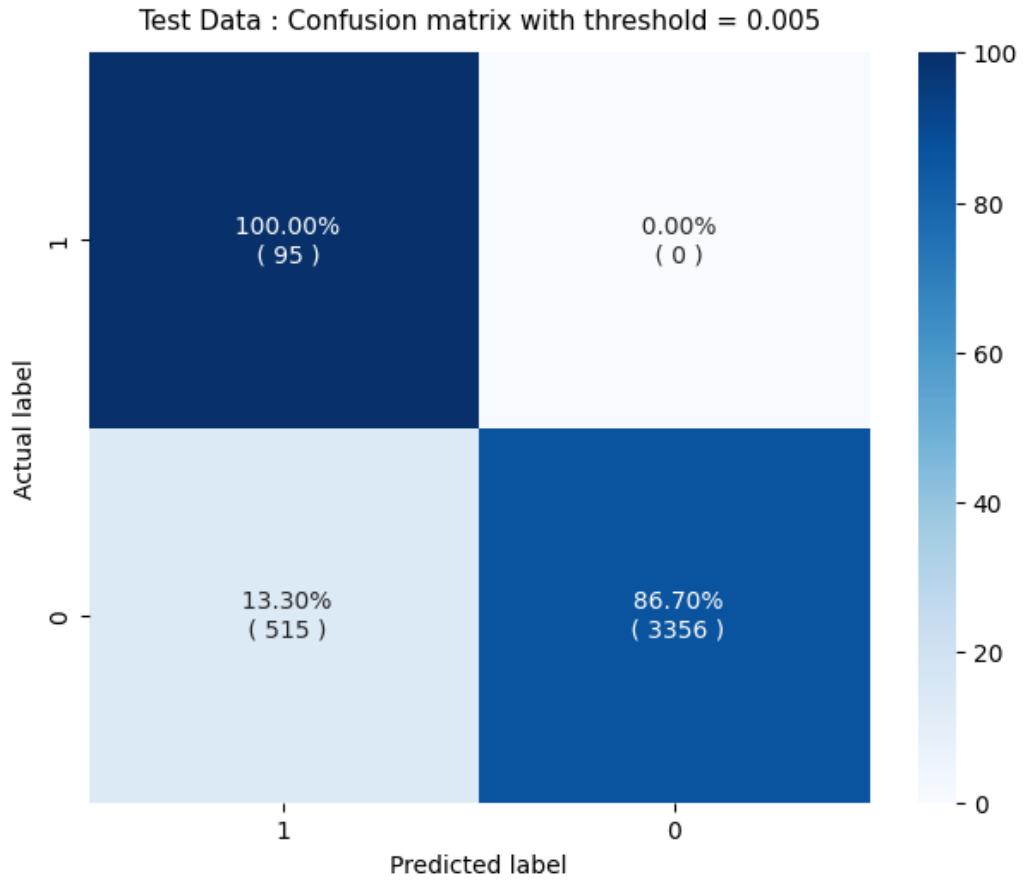


5. IMPROVED RESULTS OF BEST MODEL (AFTER IMPROVEMENT)

DATA PARAMETERS: (WR:1, PCA:TRUE, L:45, DT: 10)

THRESHOLD: 0.005

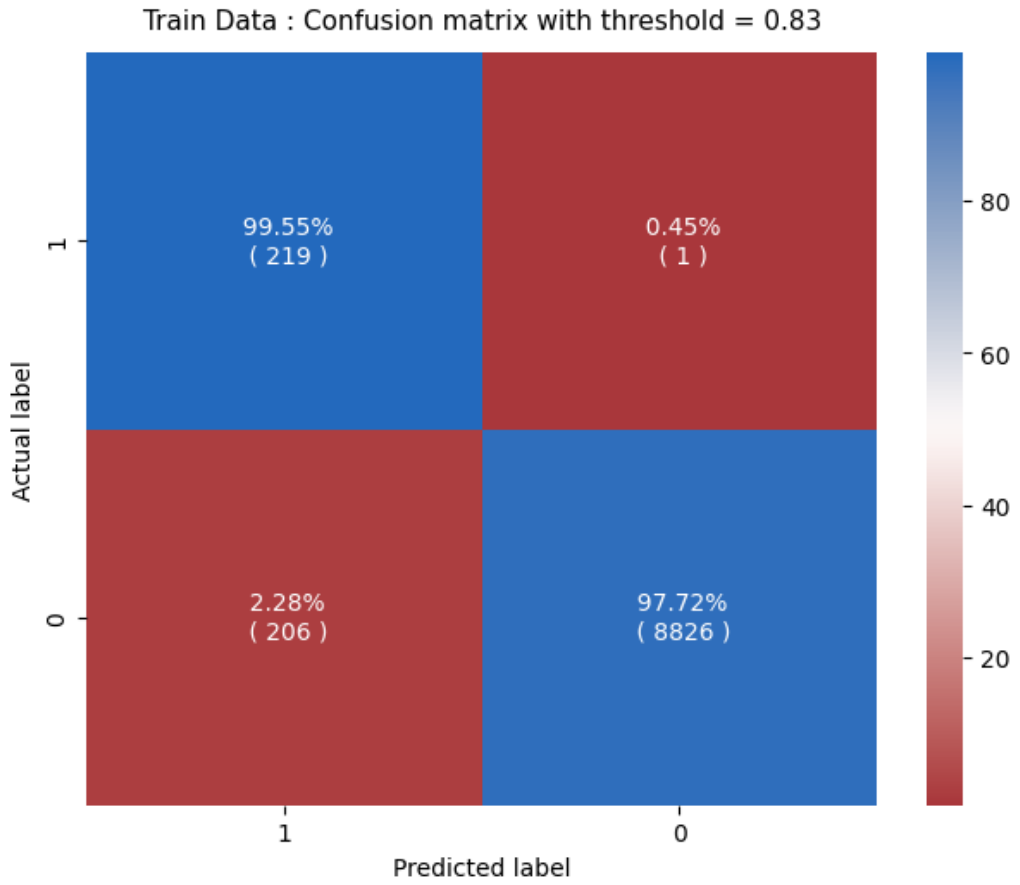


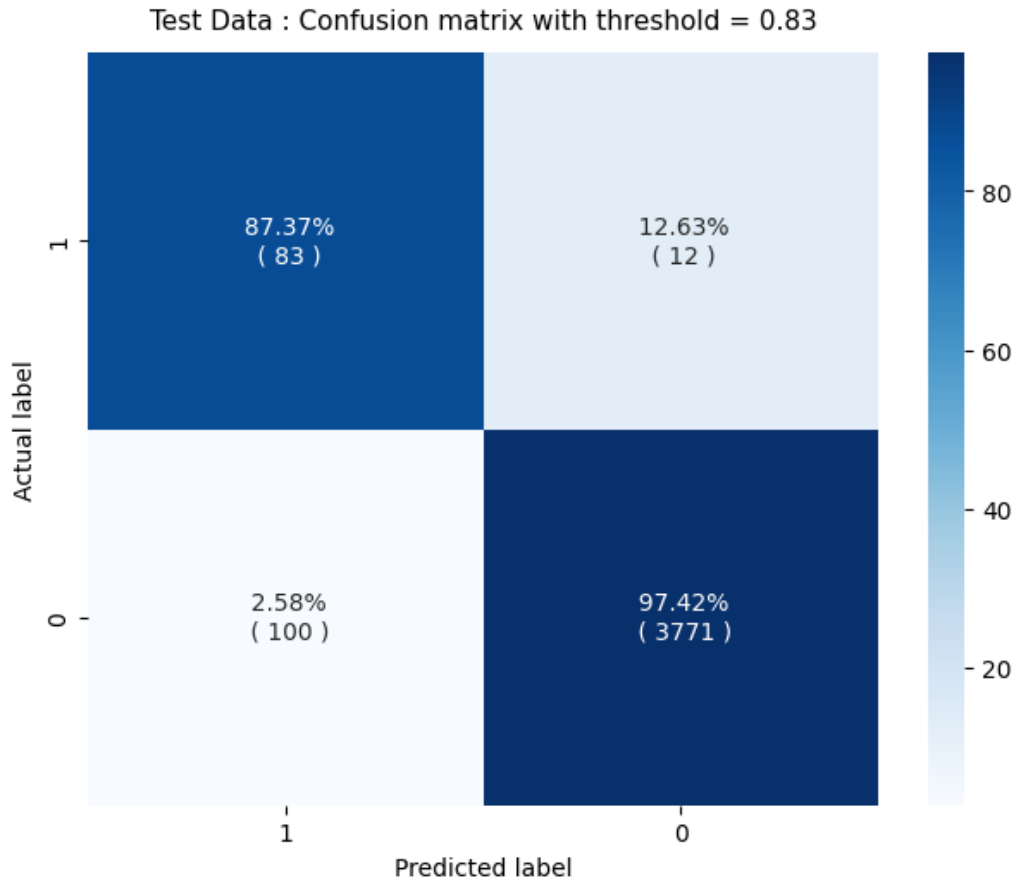


5. IMPROVED RESULTS OF BEST MODEL (AFTER IMPROVEMENT)

DATA PARAMETERS: (WR:1, PCA:TRUE, L:45, DT: 10)

THRESHOLD: 0.83





V.
CONCLUSION

CONCLUSION

- We have successfully found a good model that is able to predict an incoming crash in a time-series
- Faced with challenges like highly imbalanced data and dead solutions, we utilized a variety of methods (PCA, Stratification, Class weights, etc.) to overcome them.
- Doing Grid Search with models and parameters enabled us to find out the optimal model and parameters suitable for this task. With the help of metrics and plots, we were able to quantify and visualize the comparisons.
- Transformers with PCA work best along with the ability to choose Thresholds and WR, which can be tuned according to practical needs.
- This problem can be extended beyond transformers by using experimental models that might show even better performance.

THANK YOU

GROUP 2315

REFERENCES

- Keras References: Time Series Classification , Text Classification
- Illustrated Guide to Transformers
- Illustrated Self Attention
- F-Beta score reference
- Data Imbalance reference