



Traditional ML

vs

MLP

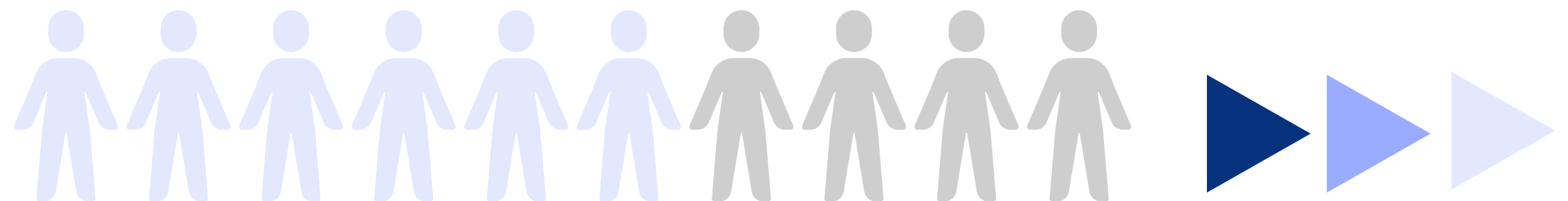
For Diabetes Binary Classification

Introduction

- This project aims to study the performance of machine learning algorithms between traditional ML and Multi-Layer Perceptron (MLP) with data set "diabetes _ binary _ health _ indicators".
- The task is Binary classification which evaluates model performance by F1-score.

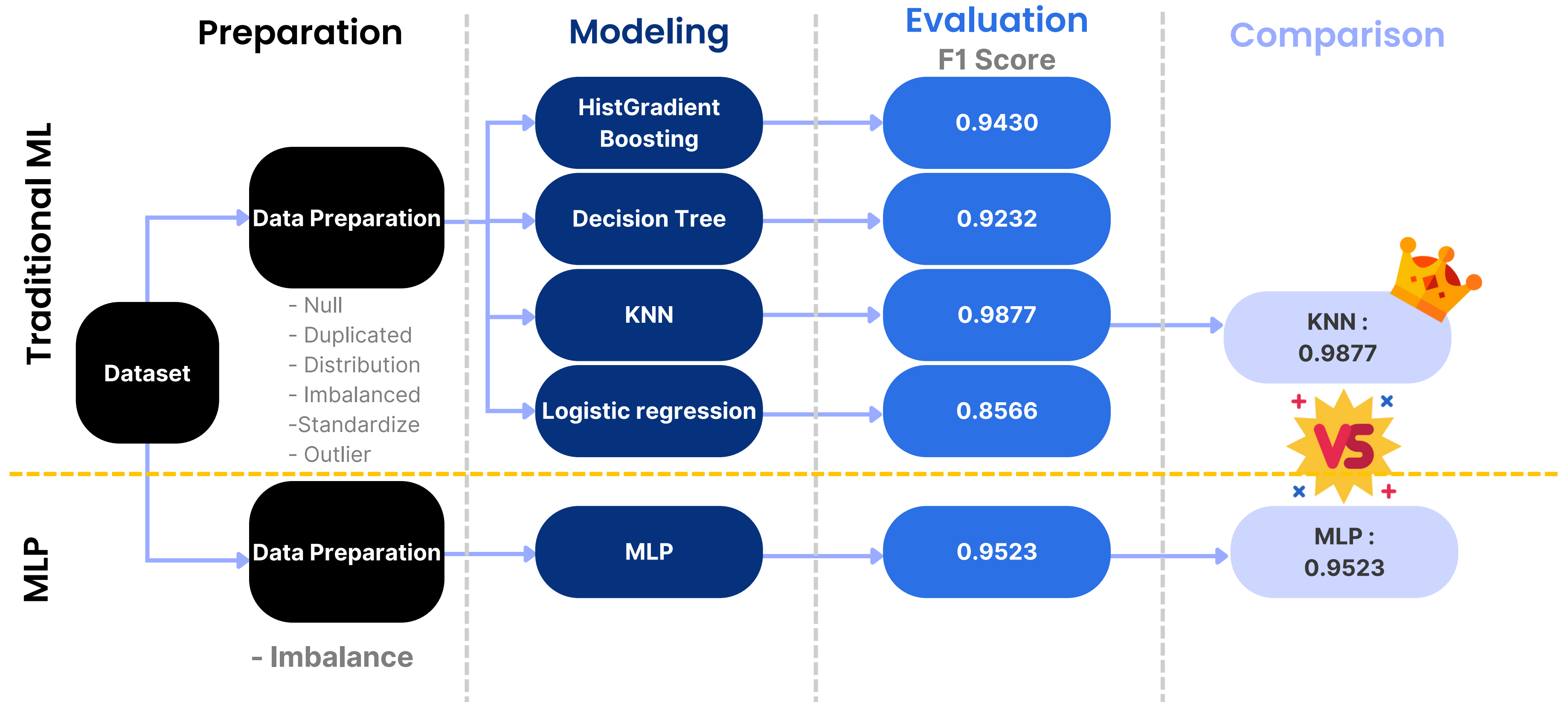
Our assumption

- As the dataset is quite big 253,680 surveys and has 21 feature variables with not balanced between diabetes and non-diabetes, we assume that MLP should show a high performance than traditional ML.



0 is for no diabetes, and 1 is for prediabetes or diabetes.

Highlight)



Highlight)

From the experiment, KNN (Traditional ML) was better than MLP when the model effect was measured with F1 score.

KNN's F1 value is 0.9878 and MLP's F1 value is 0.9523

In this dataset, Traditional ML can perform slightly better than MLP.

(Our assumption is MLP should be better than traditional ML)



From the experiment, KNN (Traditional ML) was better than MLP when the model time was measured by how long it takes to complete in modeling process.

KNN's time is 0.02 sec and MLP's time is 1,266 sec

In this dataset, Traditional ML can perform better than MLP.



Data Set & EDA

About Dataset

Heart Disease is among the most prevalent chronic diseases in the United States, impacting millions of Americans each year and exerting a significant financial burden on the economy. In the United States alone, heart disease claims roughly 647,000 lives each year — making it the leading cause of death. The buildup of plaques inside larger coronary arteries, molecular changes associated with aging, chronic inflammation, high blood pressure, and diabetes are all causes of and risk factors for heart disease.

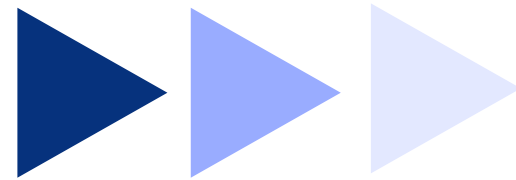
Ref: <https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset>

Dataset Column

Diabetes_binary	HvyAlcoholConsump
HighBP	AnyHealthcare
HighChol	NoDocbcCost
CholCheck	GenHlth
BMI	MentHlth
Smoker	PhysHlth
Stroke	DiffWalk
HeartDiseaseorAttack	Sex
PhysActivity	Age
Fruits	Education
Veggies	Income

Data Preparation

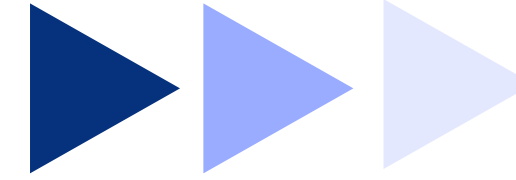
01.



Null

Diabetes_binary	0
HighBP	0
HighChol	0
CholCheck	0
BMI	0
Smoker	0
Stroke	0
HeartDiseaseorAttack	0
PhysActivity	0
Fruits	0
Veggies	0
HvyAlcoholConsump	0
AnyHealthcare	0
NoDocbcCost	0
GenHlth	0
MentHlth	0
PhysHlth	0
DiffWalk	0
Sex	0
Age	0
Education	0
Income	0
dtype: int64	

02.

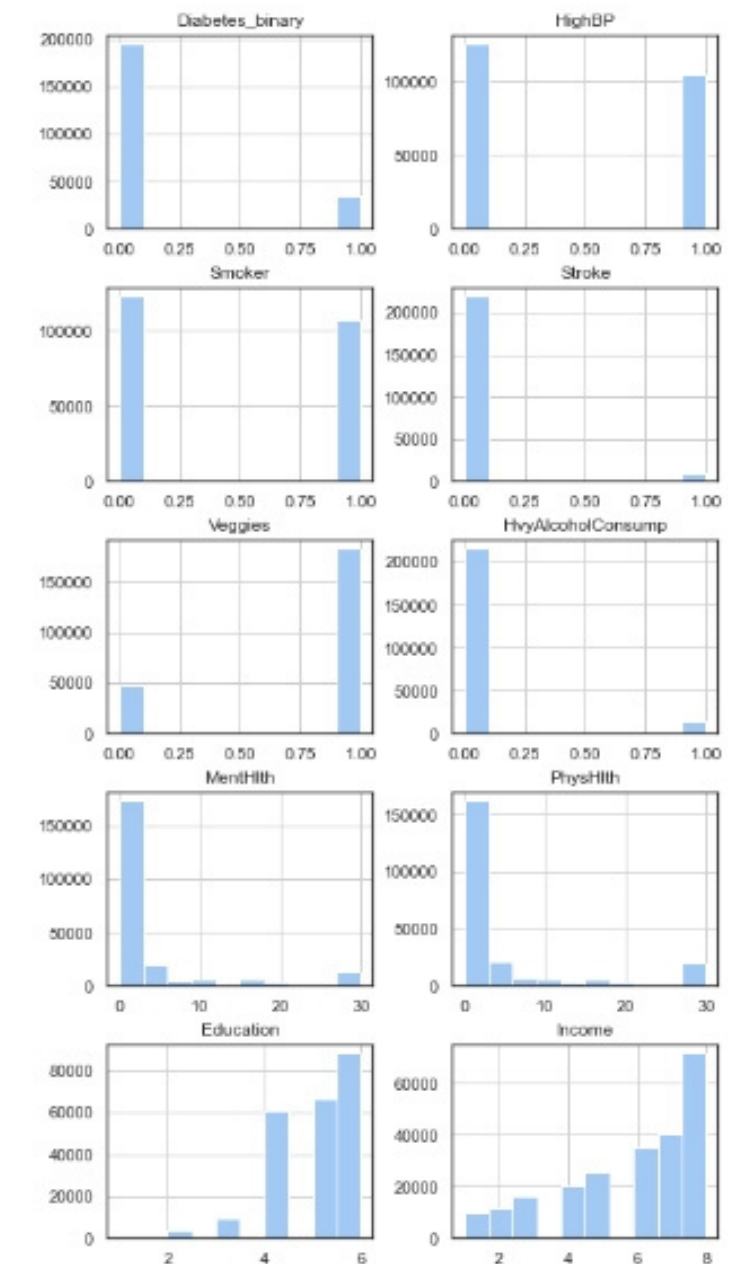


Duplicated

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 229474 entries, 0 to 253679
Data columns (total 22 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Diabetes_binary                       229474 non-null float64
1   HighBP                               229474 non-null float64
2   HighChol                             229474 non-null float64
3   CholCheck                            229474 non-null float64
4   BMI                                   229474 non-null float64
5   Smoker                               229474 non-null float64
6   Stroke                               229474 non-null float64
7   HeartDiseaseorAttack                 229474 non-null float64
8   PhysActivity                         229474 non-null float64
9   Fruits                               229474 non-null float64
10  Veggies                              229474 non-null float64
11  HvyAlcoholConsump                   229474 non-null float64
12  AnyHealthcare                       229474 non-null float64
13  NoDocbcCost                         229474 non-null float64
14  GenHlth                             229474 non-null float64
15  MentHlth                            229474 non-null float64
16  PhysHlth                            229474 non-null float64
17  DiffWalk                            229474 non-null float64
18  Sex                                  229474 non-null float64
19  Age                                  229474 non-null float64
20  Education                            229474 non-null float64
21  Income                              229474 non-null float64
dtypes: float64(22)
```

03.

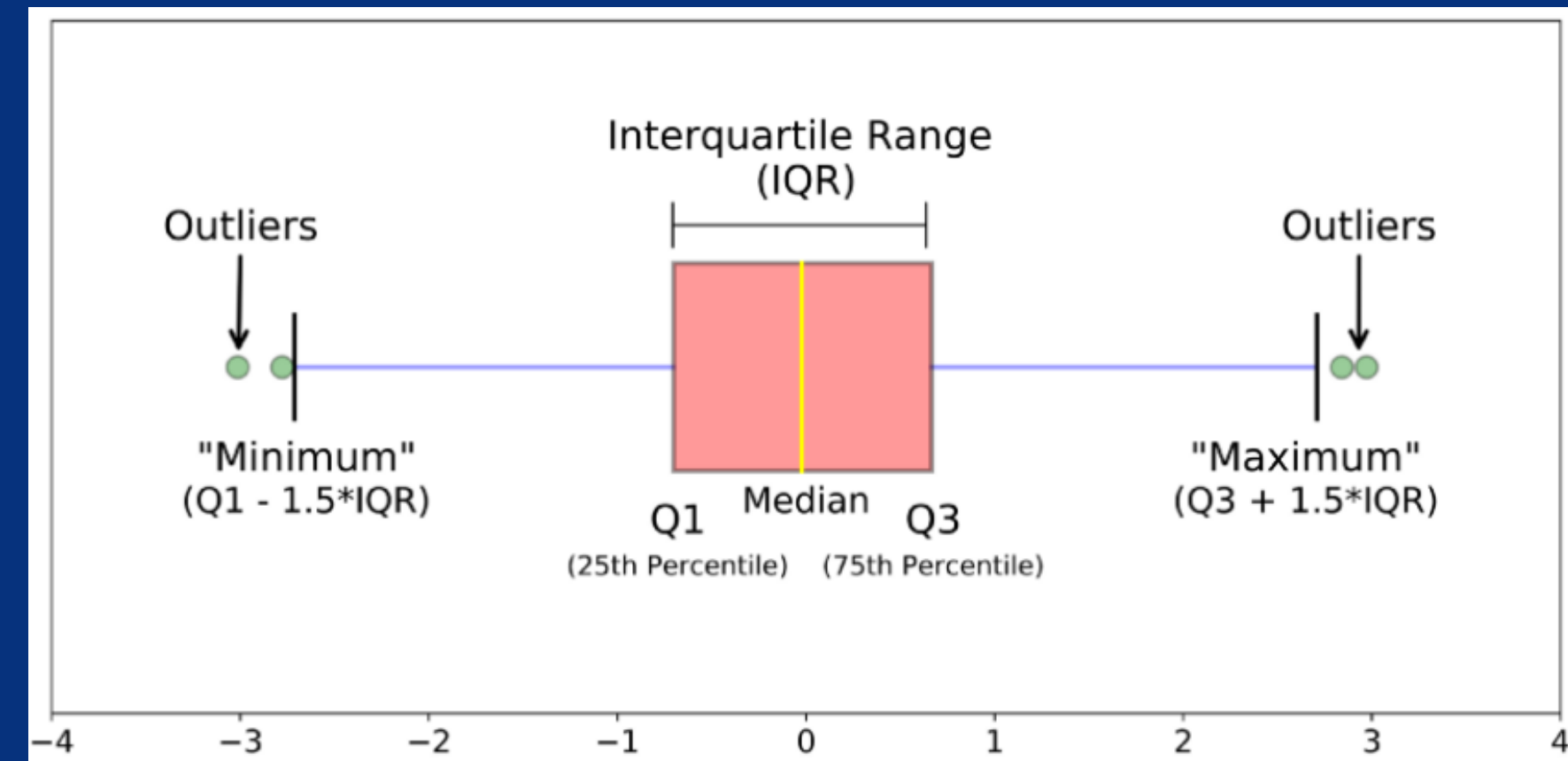
Distribution



Data Preparation

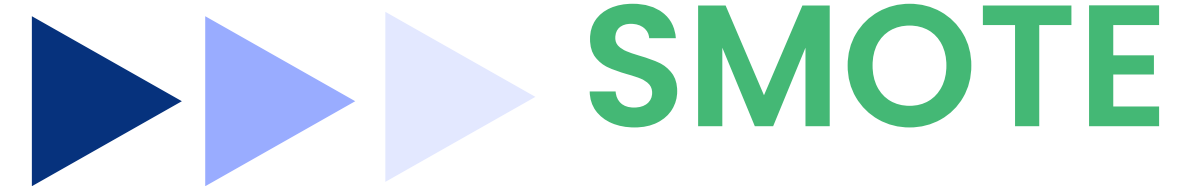
Handle Outlier ►►► IQR (Interquartile range)

The interquartile range is often used to find **outliers in data**. Outliers here are defined as observations that fall below $Q1 - 1.5 \text{ IQR}$ or above $Q3 + 1.5 \text{ IQR}$. In a boxplot, the highest and lowest occurring value within this limit are indicated by whiskers of the box (frequently with an additional bar at the end of the whisker) and any outliers as individual points.

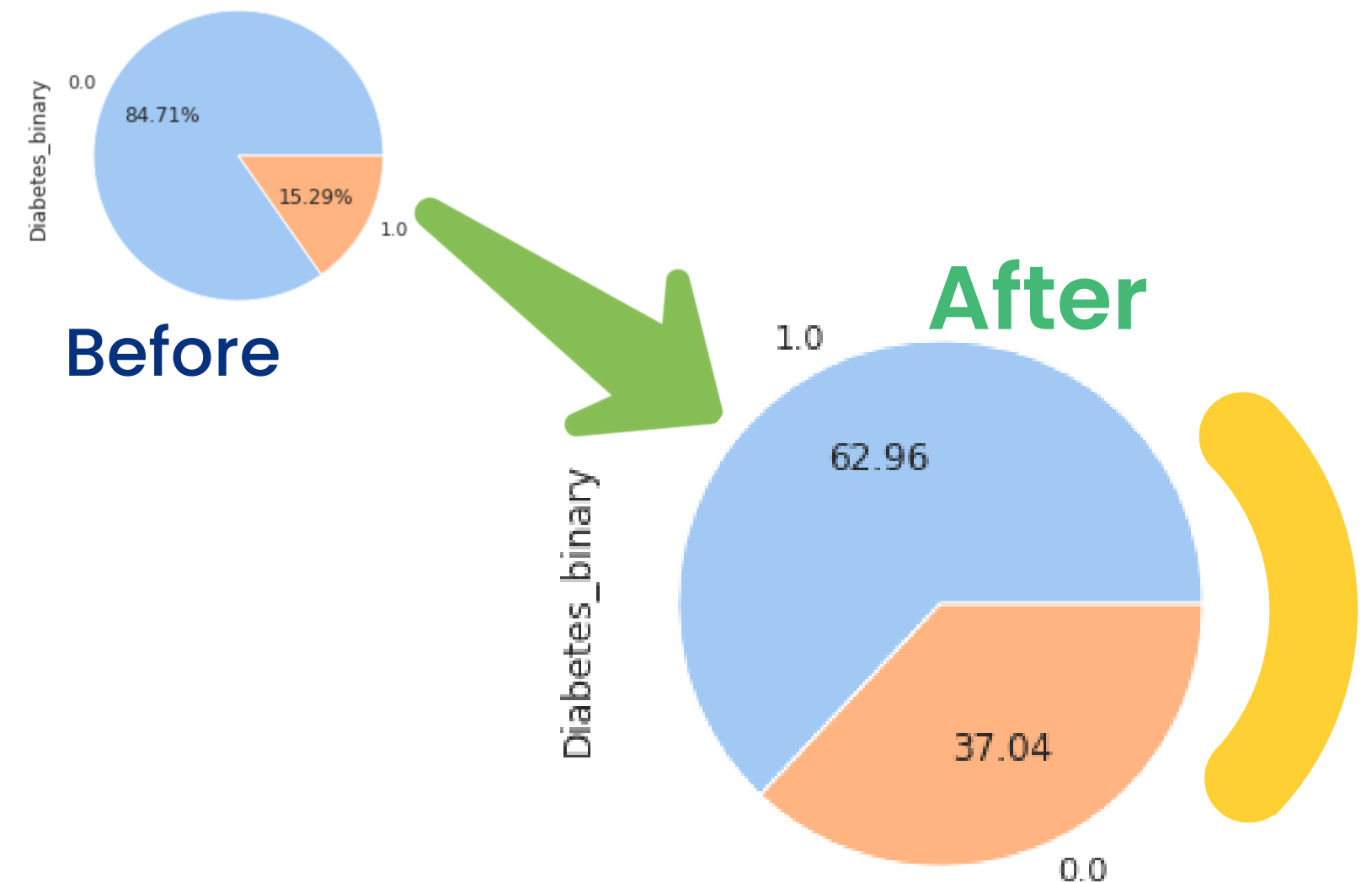


Data Preparation

Handle Imbalance



We previously presented SMOTE and showed that this method can generate **noisy samples** by interpolating new points between marginal outliers and inliers. This issue can be solved by cleaning the space resulting from over-sampling.



Data Preparation

Handle Standardize ►►► StandardScaler

Standardize features by removing the mean and scaling to unit variance.

Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. **Mean and standard deviation are then stored to be used on later data using transform.**

Data Preparation

Extract Features ►►► **Features selection**

Feature ranking with recursive feature elimination.

Given an external estimator that assigns weights to features, the goal of recursive feature elimination is to select features by recursively considering smaller and smaller sets of features.

Traditional ML

Data Preparation



	HistGradientBoosting	Dicision Tree	KNN	Logistic Regression
Outlier				
Normalize				
Imbalance "Y"				
Feature Selection				



Traditional ML

HistGradientBoosting , Decision Tree , KNN , Logistic regression

(Selected Models

01.

HistGradient
Boosting

02.

Decision Tree

03.

K-Nearest
Neighbors
(KNN)

04.

Logistic
regression

Model Evaluation)

Accuracy (ACC.)

Model accuracy is the measurement used to determine which model is best at identifying relationships and patterns between variables in a dataset based on the input, or training, data.

Precision

What proportion of positive identifications was actually correct?

Recall

What proportion of actual positives was identified correctly?

F1

By definition, F1-score is the harmonic mean of precision and recall. It combines precision and recall into a single number



HistGradientBoosting)

Outlier
IQR

Normalize

Imbalance "Y"
SMOTE_ENN

Feature
Selection

Split

80/20

ACC.

0.9429

Precision

0.9433

Recall

0.9429

F1

0.9430

Dicision Tree)

Outlier
IQR

Normalize

Imbalance "Y"
SMOTE_ENN

Feature
Selection

Split	80/20
ACC.	0.9231
Precision	0.9232
Recall	0.9231
F1	0.9232

KNN)

Outlier
IQR

Normalize

Imbalance "Y"
SMOTE_ENN

Feature
Selection

Split

80/20

ACC.

0.9878

Precision

0.9879

Recall

0.9878

F1

0.9877

Logistic Regression)

Outlier
IQR

Normalize

Imbalance "Y"
SMOTE_ENN

Feature
Selection

Split

ACC.

Precision

Recall

F1

70/30

0.8576

0.8565

0.8576

0.8566

Traditional ML

Conclusion

	HistGradientBoosting	Dicision Tree	KNN 	Logistic Regression
Split	80/20	80/20	80/20	70/30
ACC.	0.9429	0.9231	0.9878	0.8576
Precision	0.9433	0.9232	0.9879	0.8565
Recall	0.9429	0.9231	0.9878	0.8576
F1	0.9430	0.9232	0.9877	0.8566

KNN is the Best



MLP

(How We Trial And Error

(Number of Hidden Layer

Try to increase number of hidden layer from 2 to see what happen.

Number of BatchNormalization Layer

Try to increase number of BatchNormalization layer to see what happen.

Dropout Rate

Try to increase and decrease number of dropout rate between 0.1-0.2

Number of Node

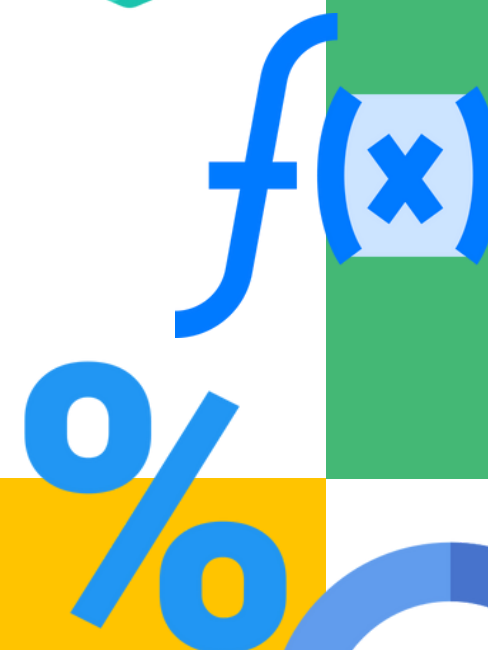
Try to increase number of node in each hidden layer 32, 64, 128, 512, and 1024 to see which combination did the best performance

Activation Function in Eeach Hidden Layer / Output Layer

Each Hidden Layer :Try to use ReLU, Tanh
Output Layer : Softmax, Sigmoid

Number of Epochs

If F1 has an uptrend we try to increase the number of epoch




Trial And Error)

(Fix Hyperparameter

Type of the optimizer :	Adam
Learning rate :	0.0001
Batch size :	512
Validation split :	80/20

Top 10 Combination
sort by Accuracy



	Number of Hidden Layer	Number of Node	Number of BatchNorm Layer	Act. Func. (Hidden Layer)	Act. Func. (Output Layer)	Number of Epochs	Accuracy (test set)	Runtime (min.)
1	 6	512, 1024, 64, 32, 512, 512	6	ReLU	Sigmoid	300	0.9528	27.15
2	6	512, 1024, 64, 32, 512, 512	6	ReLU	Softmax	300	0.9509	17.25
3	6	512, 1024, 64, 32, 512, 512	6	ReLU	Sigmoid	200	0.9492	16.51
4	2	512, 512	2	Tanh	Sigmoid	1000	0.9460	38.23
5	6	512, 1024, 64, 32, 512, 512	6	ReLU	Sigmoid	100	0.9430	7.24
6	2	256, 512	2	Tanh	Sigmod	500	0.9427	21.56
7	6	512, 1024, 64, 32, 512, 512	6	ReLU	Sigmod	100	0.9426	7.39
8	5	512, 1024, 64, 32, 512	5	ReLU	Sigmoid	100	0.9404	7.46
9	6	512, 1024, 64, 32, 512, 256	6	ReLU	Sigmod	100	0.9402	9.43
10	5	512, 1024, 64, 32, 512	5	ReLU	Sigmod	100	0.9387	5.24

MLP Training

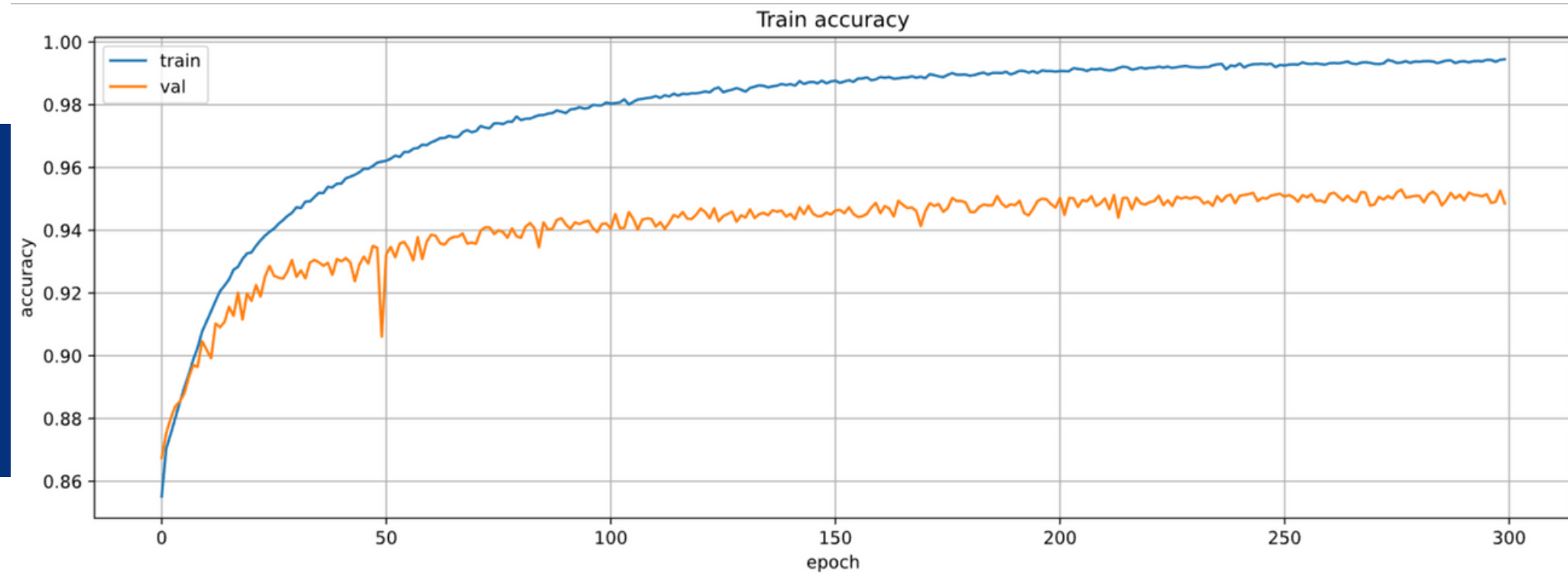
Number of hidden layers	6
Number of nodes	512, 1024, 64, 32, 512, 512
Type of activation function hidden layer	relu
Batch Norm	6
Number of nodes in output layer	1
Type of activation function in output layer	sigmoid
Drop out rate	0.2

Type of the optimizer	adam
Learning rate	0.0001
Type of the loss function	BinaryCrossentropy
Type of evaluation metric(s)	acc
Batch size	512
Number of epochs	300
Validation split	0.2

Traditional ML VS MLP)

	HistGradientBoo sting	Dicision Tree	 KNN 	Logistic Regression	MLP (avg. 5 times)
Split	80/20	80/20	80/20	70/30	80/20
ACC.	0.9429	0.9231	0.9878	0.8576	0.9525
Precision	0.9433	0.9232	0.9879	0.8565	0.9526
Recall	0.9429	0.9231	0.9878	0.8576	0.9525
F1	0.9430	0.9232	0.9877	0.8566	0.9523
Runtime (sec.)	12.97	4.35	0.02	3.72	1,266

Train vs Validation



F1 score 0.9523 ± 0.001
(5 times)

Train
Accuracy



Train
Loss



(Train accuracy and train loss show overfit result)



Discussion

The imbalance correction


increased Accuracy in both Traditional ML and MLP, as expected. In addition, an imbalance with SMOTE_ENN (from imblearn.combine import SMOTEENN) resulted in a more significant increase in Accuracy. Compared to sklearn oversampling (from sklearn.utils import resample).

The increasing number of layers

does not guarantee that the Accuracy will increase. Because if we set a small number of layers, but if there is a large number of epochs, it can increase the Accuracy.



Discussion



The Epoch number affects the Accuracy,


as expected. But if we increase the number of Epoch more and more. Found that it doesn't affect Accuracy, which is unexpected because the team understands that it can keep pushing until the Accuracy reaches 0.99.

The number of batch sizes had no significant effect on Accuracy,

which was unexpected as the team understood that more batch sizes would increase Accuracy.

The higher the learning rate,

the lower the default value. This increased the Accuracy, as expected. Initially, the team used default 0.01, then adjusted it to 0.0001.





Conclusiön



Conclusion

Summarize the results of this project

1. Adjusting Hyperparameter between Traditional ML and MLP

- Adjusting the Hyperparameter of MLP is more diverse and complex than traditional ML. Hyperparameter is more difficult, as most of the MLP research teams did not provide clear principles for defining the Network architecture, but using the Trial and Error principle, the team decided to use the Trial and Error method, so the results of the MLP could still be improved.

2. Time and Resources to Build the Model Between Traditional ML and MLP

- Due to the team using Trial and Error principles, it takes much more time and resources to build Model MLP (Extremely) Traditional ML, even though the data is small and tabular. Therefore, tasks that are not complicated should use Traditional ML. More than an MLP.



Reference

Basic machine learning

- youtube chanel: data professor
(Associate Professor Chanin Nantasenamat, Ph.D.)

Handle outlier

- <https://bit.ly/Handle-outlier-1>
- <https://bit.ly/Handle-outlier-2>

AE-MLP

- A Hybrid Deep Learning Approach for DDoS Detection and Classification, Yuanyuan Wei; Julian Jang-Jaccard; Fariza Sabrina; Amardeep Singh; Wen Xu; Seyit Camtepe, IEEE Access (Volume: 9), Page(s): 146810 – 146821, 27 October 2021

Reference version :)

pandas	1.3.5
numpy	1.21.5
matplotlib	3.2.2
seaborn	0.12
sklearn	1.1.2
TensorFlow	2.9.1
Imbalanced-learn	0.9.1
GPU	<u>NVIDIA GeForce RTX 3060 Laptop GPU</u> (UUID: GPU-2dd56641-cce9-ed20-18ec-e52ad1b56cd3)



Deepsleep's Member :)

- **Research**
- **Train and tune ML**
- **Write result, discussion, conclusion report**

(20%) Athit Santikarn_6410414007
<https://github.com/zoomthebear>

(20%) Suphanun Sukamta_6410422020
<https://github.com/ssuphanun>

- **Prepare and clean dataset**
- **Train and tune ML & MLP model**
- **Write network architecture**
- **Training**
- **Discussion, Conclusion report**

(20%) Chokchai Kenpho_6410422004
<https://github.com/Bolympus1>

(20%) Noppol Anakpluek_6410422009
<https://github.com/noppolanak>

(20%) Watcharakorn Pasanta_6420422006
<https://github.com/WatcharakorP>



 This project is a part of Course
"DADS7202 Deep Learning"
Data Analytics and Data Science, NIDA.

Thank you :)

Back up

Table1: MLP Trial & Error 39 combinations

No#	Max val_acc	Preparation	Number of hidden layers in the network	Number of nodes in each hidden layer	Type of activation function in each hidden layer	Batch Norm	Number of nodes in output layer	Type of activation function in output layer	Drop out rate	Type of the optimizer	Learning rate	Type of the loss function	Type of evaluation metric(s)	Batch size	Number of epochs	Validation split	Accuracy	Precision	Recall	F1-Score	Run time (min)
1	0.9057	smote_ENN	3	32 64 32	relu	3	2	softmax	0.3	adam	default	sparse_categorical_crossentropy	acc	128	20	0.2	0.9057				
2	0.9046	smote_ENN	3	32 64 32	relu	2	2	softmax	0.3	adam	default	sparse_categorical_crossentropy	acc	128	20	0.2	0.9046				
3	0.9079	smote_ENN	3	32 64 32	relu	2	2	softmax	0.1	adam	default	sparse_categorical_crossentropy	acc	128	20	0.2	0.9079				
4	0.9091	smote_ENN	3	32 64 32	relu	2	2	softmax	0.1	adam	default	sparse_categorical_crossentropy	acc	128	40	0.2	0.9091				
5	0.9191	smote_ENN	3	32 64 32	relu	2	2	softmax	0.1	adam	default	sparse_categorical_crossentropy	acc	128	80	0.2	0.9191				
6	0.6097	smote_ENN	3	32 64 32	relu	2	1	softmax	0.1	adam	default	BinaryCrossentropy	acc	128	20	0.2	0.6097				
7	0.3902	smote_ENN	3	32 64 32	relu	2	1	sigmoid	0.1	adam	default	sparse_categorical_crossentropy	acc	128	20	0.2	0.3902				
8	0.9074	smote_ENN	3	32 64 32	relu	2	1	sigmoid	0.1	adam	default	BinaryCrossentropy	acc	128	20	0.2	0.9074				
9	0.9180	smote_ENN	3	32 64 32	relu	2	1	sigmoid	0.1	adam	default	BinaryCrossentropy	acc	128	40	0.2	0.9180				
10	0.9233	smote_ENN	3	32 64 32	relu	2	1	sigmoid	0.1	adam	default	BinaryCrossentropy	acc	128	80	0.2	0.9233				
11	0.9214	smote_ENN	3	32 64 32	relu	2	1	sigmoid	0.1	adam	default	BinaryCrossentropy	acc	128	100	0.2	0.9214				
12	0.9204	smote_ENN	3	32 64 32	relu	2	1	sigmoid	0.1	adam	default	BinaryCrossentropy	acc	256	80	0.2	0.9204				
13	0.9180	smote_ENN	3	32 64 32	relu	2	1	sigmoid	0.1	adam	default	BinaryCrossentropy	acc	32	80	0.2	0.9180				
14	0.9111	smote_ENN	3	32 64 32	tanh	2	1	sigmoid	0.1	adam	default	BinaryCrossentropy	acc	128	20	0.2	0.9111				
15	0.9180	smote_ENN	3	32 64 32	tanh	2	1	sigmoid	0.1	adam	default	BinaryCrossentropy	acc	128	40	0.2	0.9180				
16	0.9215	smote_ENN	3	32 64 32	tanh	2	1	sigmoid	0.1	adam	default	BinaryCrossentropy	acc	128	80	0.2	0.9215				
17	0.9227	smote_ENN	3	32 64 32	tanh	2	1	sigmoid	0.1	adam	default	BinaryCrossentropy	acc	128	100	0.2	0.9227				
18	0.9258	smote_ENN	3	32 64 32	tanh	2	1	sigmoid	0.1	adam	default	BinaryCrossentropy	acc	256	100	0.2	0.9258				
19	0.9243	smote_ENN	3	32 64 32	tanh	2	1	sigmoid	0.1	adam	default	BinaryCrossentropy	acc	256	200	0.2	0.9243				
20	0.9185	smote_ENN	3	32 64 32	tanh	2	1	sigmoid	0.1	adam	default	BinaryCrossentropy	acc	512	100	0.2	0.9185				
21	0.9345	smote_ENN	5	512 128 64 64 64	tanh	2	1	sigmoid	0.1	adam	default	BinaryCrossentropy	acc	256	100	0.2	0.9345				
22	0.9080	smote_ENN	5	512 128 64 64 64	tanh	2	1	sigmoid	0.1	sgd	default	BinaryCrossentropy	acc	256	100	0.2	0.9080				
23	0.9169	smote_ENN	5	512 128 64 64 64	tanh	2	1	tanh	0.1	adam	default	BinaryCrossentropy	acc	256	100	0.2	0.9169				
24	0.9224	smote_ENN	2	512 512	relu	2	1	sigmoid	0.2	adam	0.0001	BinaryCrossentropy	acc	512	100	0.2	0.9224				
25	0.9322	smote_ENN	2	512 1024	relu	2	1	sigmoid	0.2	adam	0.0001	BinaryCrossentropy	acc	512	100	0.2	0.9322				
26	0.9385	smote_ENN	3	512 1024 64	relu	3	1	sigmoid	0.2	adam	0.0001	BinaryCrossentropy	acc	512	100	0.2	0.9385				
27	0.9401	smote_ENN	3	512 1024 128	relu	3	1	sigmoid	0.2	adam	0.0001	BinaryCrossentropy	acc	512	100	0.2	0.9401				
28	0.9385	smote_ENN	4	512 1024 64 32	relu	4	1	sigmoid	0.2	adam	0.0001	BinaryCrossentropy	acc	512	100	0.2	0.9385				
29	0.9414	smote_ENN	5	512 1024 64 32 64	relu	5	1	sigmoid	0.2	adam	0.0001	BinaryCrossentropy	acc	512	100	0.2	0.9414				
30	0.9426	smote_ENN	5	512 1024 64 32 128	relu	5	1	sigmoid	0.2	adam	0.0001	BinaryCrossentropy	acc	512	100	0.2	0.9388	0.9395	0.9388	0.9383	20.20
31	0.9418	smote_ENN	5	512 1024 64 32 512	relu	5	1	sigmoid	0.2	adam	0.0001	BinaryCrossentropy	acc	512	100	0.2	0.9405	0.9404	0.9405	0.9402	7.46
32	0.9426	smote_ENN	6	512 1024 64 32 512 128	relu	6	1	sigmoid	0.2	adam	0.0001	BinaryCrossentropy	acc	512	100	0.2	0.9426	0.9427	0.9426	0.9423	7.39
33	0.9428	smote_ENN	6	512 1024 64 32 512 256	relu	6	1	sigmoid	0.2	adam	0.0001	BinaryCrossentropy	acc	512	100	0.2	0.9402	0.9404	0.9402	0.9398	9.43
34	0.9448	smote_ENN	6	512 1024 64 32 512 512	relu	6	1	sigmoid	0.2	adam	0.0001	BinaryCrossentropy	acc	512	100	0.2	0.9430	0.9429	0.9430	0.9428	7.24
35	0.9497	smote_ENN	6	512 1024 64 32 512 512	relu	6	1	sigmoid	0.2	adam	0.0001	BinaryCrossentropy	acc	512	200	0.2	0.9492	0.9494	0.9492	0.9490	16.51
36	0.9527	smote_ENN	6	512 1024 64 32 512 512	relu	6	1	sigmoid	0.2	adam	0.0001	BinaryCrossentropy	acc	512	300	0.2	0.9528	0.9531	0.9528	0.9526	27.15
37	0.9510	smote_ENN	6	512 1024 64 32 512 512	relu	6	2	softmax	0.2	adam	0.0001	sparse_categorical_crossentropy	acc	512	300	0.2	0.9509	0.9510	0.9509	0.9507	17.25
38	0.9429	smote_ENN	2	256 512	tanh	2	1	sigmoid	0.1	adam	0.0001	BinaryCrossentropy	acc	512	500	0.2	0.9427	0.9428	0.9427	0.9425	21.56
39	0.9536	smote_ENN	2	512 512	tanh	2	1	sigmoid	0.1	adam	0.0001	BinaryCrossentropy	acc	512	1000	0.2	0.9460	0.9461	0.9460	0.9458	38.23

(Back up

Table2: The Best MLP

No#	36
Max val_acc	0.9527
Preparation	smote_ENN
Number of hidden layers in the network	6
Number of nodes in each hidden layer	512 1024 64 32 512 512
Type of activation function in each hidden layer	relu
Batch Norm	6
Number of nodes in output layer	1
Type of activation function in output layer	sigmoid
Drop out rate	0.2
Type of the optimizer	adam
Learning rate	0.0001
Type of the loss function	BinaryCrossentropy
Type of evaluation metric(s)	acc
Batch size	512
Number of epochs	300
Validation split	0.2
Accuracy	0.9528
Precision	0.9531
Recall	0.9528
F1-Score	0.9526
Run time (min)	27.15

Run#	Accuracy	Precision	Recall	F1-Score	Run time (mm:ss)
1	0.9536	0.9537	0.9535	0.9534	21:29
2	0.9517	0.9520	0.9520	0.9518	22:07
3	0.9521	0.9520	0.9520	0.9519	24:58
4	0.9536	0.9538	0.9537	0.9535	19:57
5	0.9513	0.9515	0.9513	0.9511	20:25
Average	0.9525	0.9526	0.9525	0.9523	21:47
Std.dev	0.0011	0.0011	0.0010	0.0011	1:58