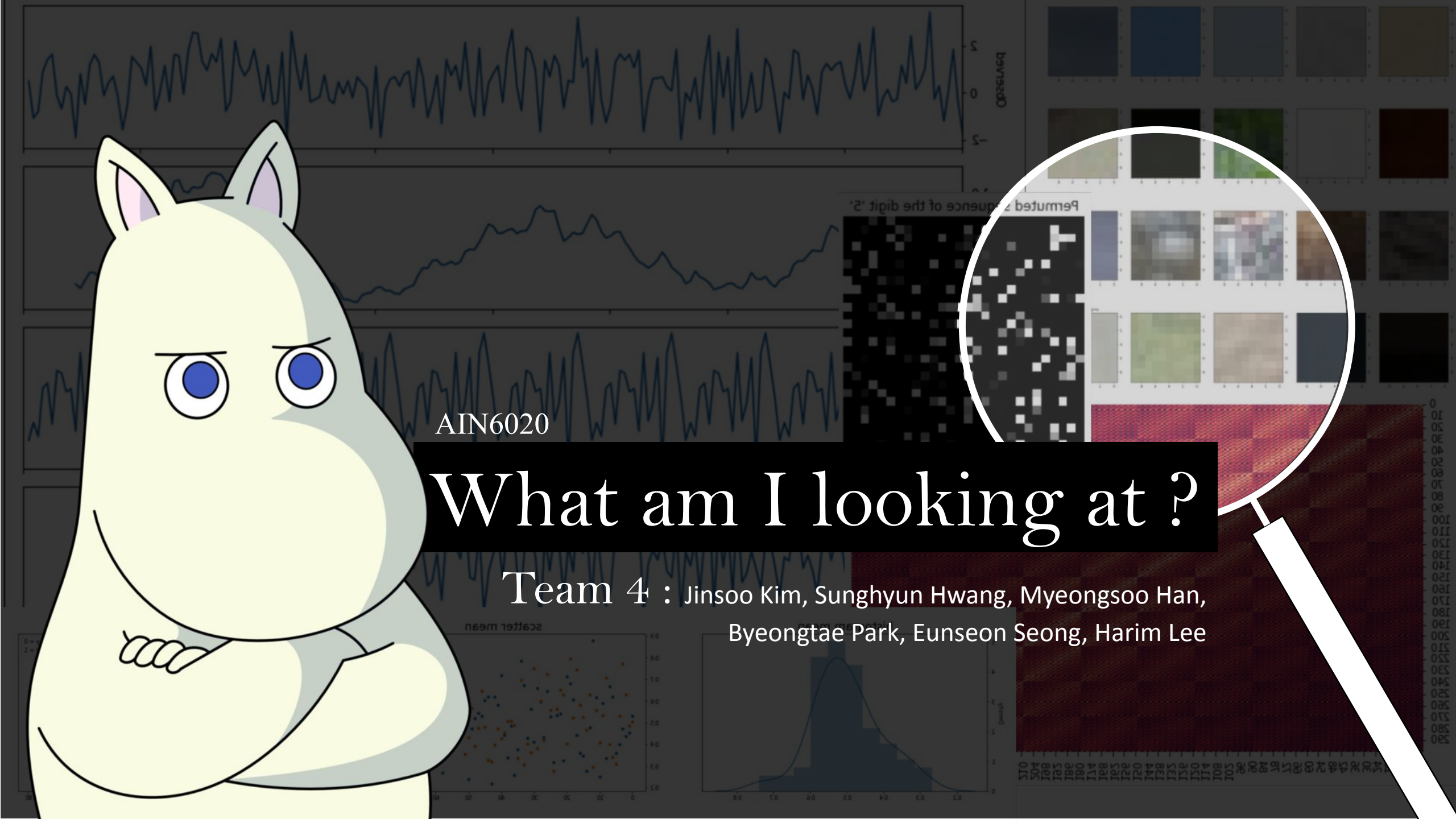


AIN6020

What am I looking at ?

Team 4 : Jinsoo Kim, Sunghyun Hwang, Myeongsoo Han,
Byeongtae Park, Eunseon Seong, Harim Lee

Dataset 1

- Exploration Data Analysis
- Modeling
- Result
- Appendix

01 Exploration Data Analysis

Dataset	Instances	Dimensions	NaN	Outlier	Categorical	Numerical	Label_0	Label_1
Train	700	200	0	49th column	0	200	500	200
Test	350	200	0	49th column	0	200	250	100

Table 1: Summary statistics of Datatset 1.

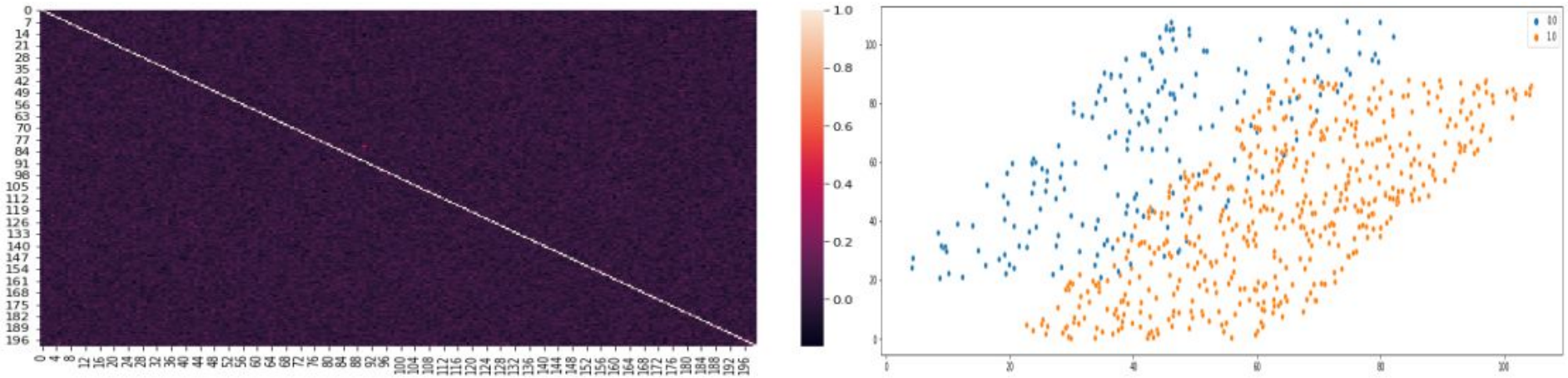


Figure 1: Heatmap of correlation on train data.

01 Exploration Data Analysis

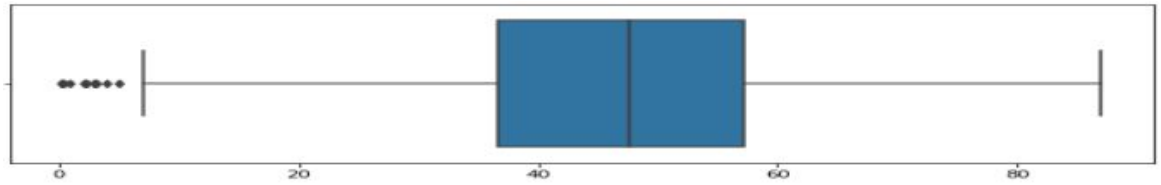


Figure 2: Box plot of 49th feature on train data.

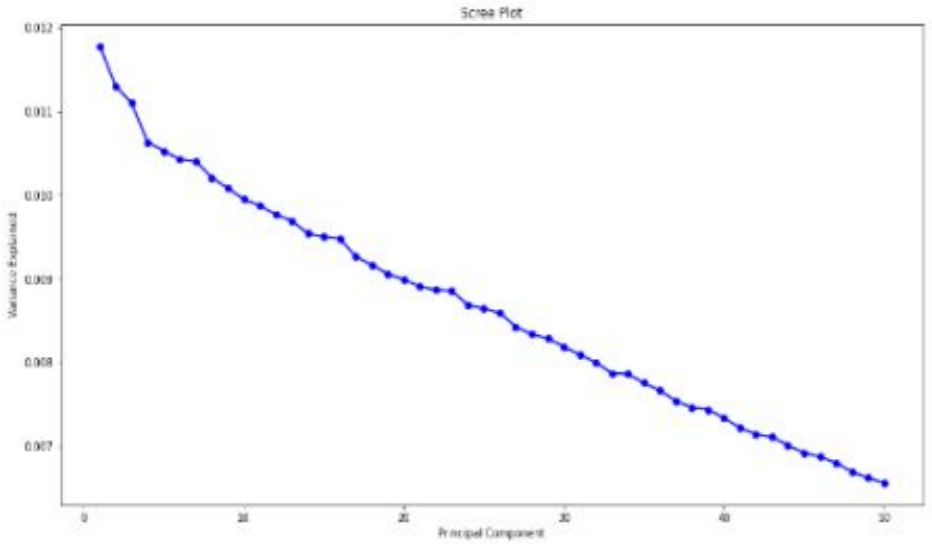


Figure 3: Scree plot of principal components on train data.

02 Modeling

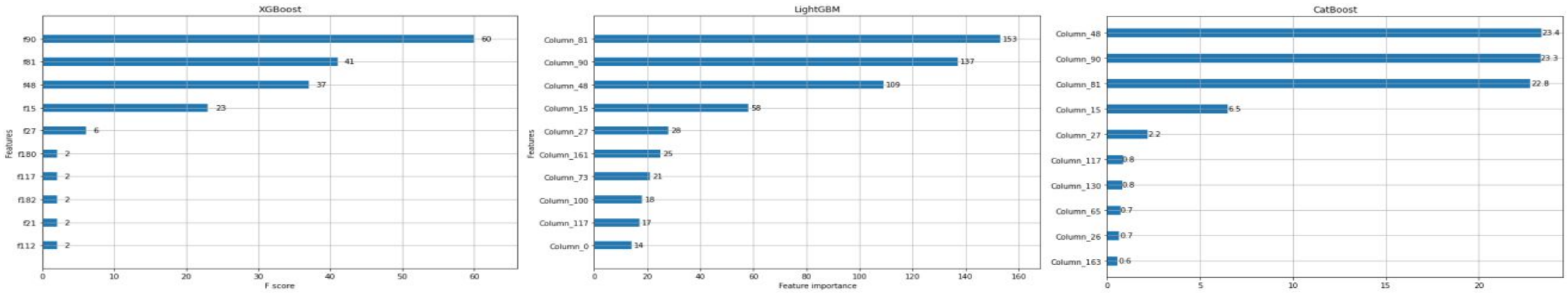


Figure 4: Feature importance of three boosting algorithms.

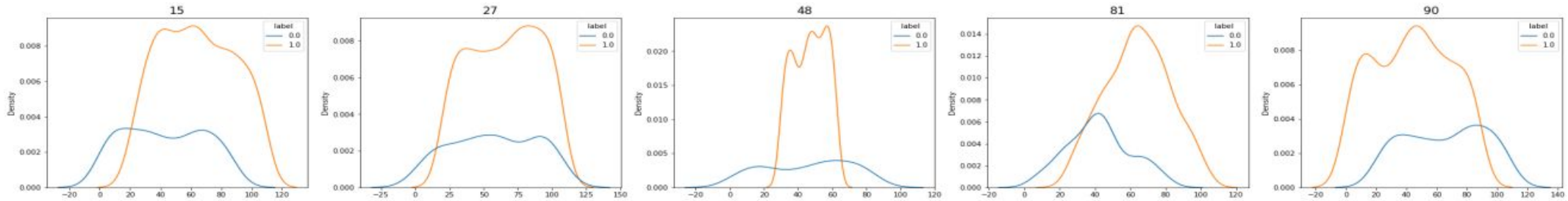


Figure 5: Kernel density estimate plot of five critical features on train data.

03 Result

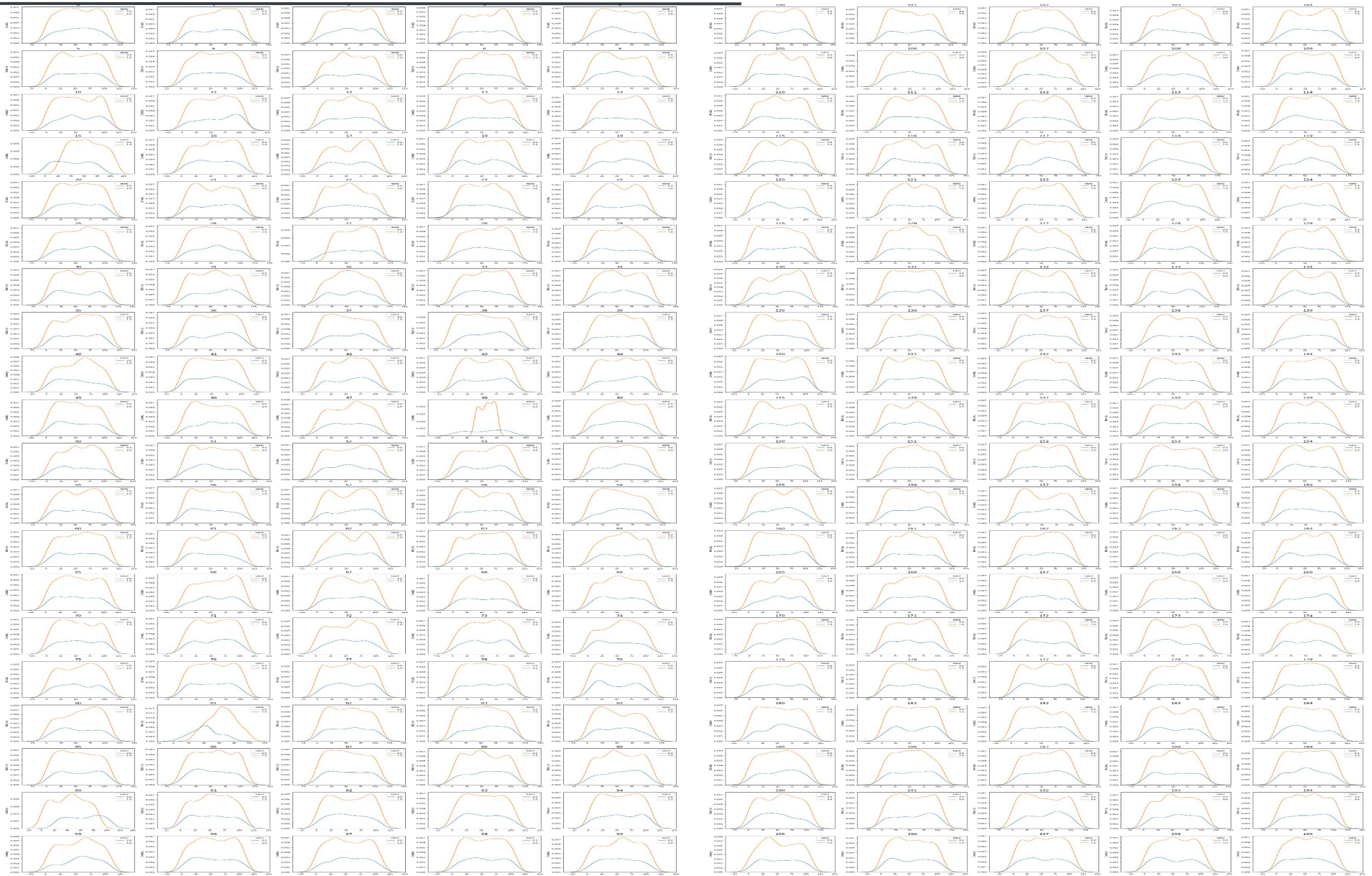
Input	XGBoost	LightGBM	CatBoost
All features	98.29%	98.57%	97.71%
Five critical features	98.57%	98.86%	99.14%

Table 2: Accuracy of three boosting algorithms.

Input	Precision	Recall	F1-score	Support
Label0	1.0	0.97	0.985	100
Label1	0.988	1.0	0.994	250
Macro avg	0.994	0.985	0.989	350
Weighted avg	0.992	0.991	0.991	350

Table 3: Classification report of best CatBoost algorithm.

04 Appendix



Dataset 2

Data Description

Binary classification with balanced label

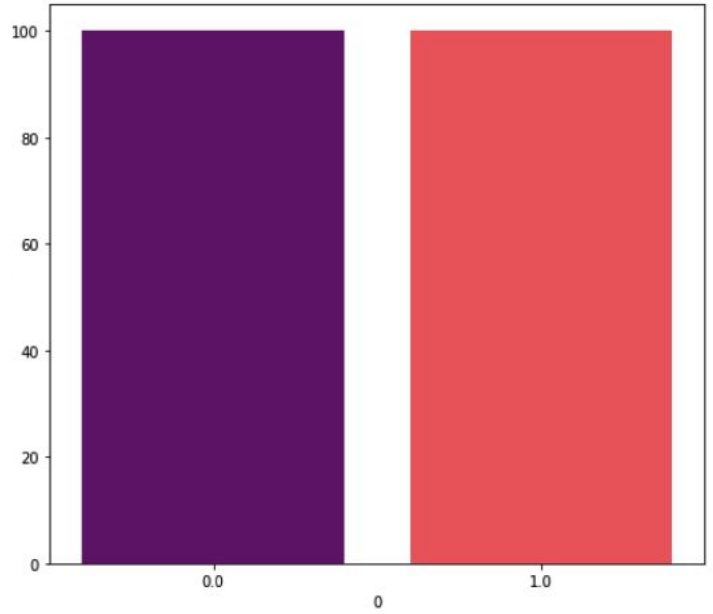
Basic Info. of Train data

```
x.shape:  (200, 80)  y.shape:  (200,)
label_info:
[0. 1.]
per-class number
0.0 : [100]
1.0 : [100]
```

Basic Info. of Test data

```
x.shape:  (400, 80)  y.shape:  (400,)
label_info:
[0. 1.]
per-class number
0.0 : [200]
1.0 : [200]
```

(a) Basic information of Dataset 2.



(b) Label ratio of Dataset 2.

Data Description

Statistics

	0	1	2	3	4	...	75	76	77	78	79
count	200.000000	200.000000	200.000000	200.000000	200.000000	...	200.000000	200.000000	200.000000	200.000000	200.000000
mean	0.517334	0.470278	0.532742	0.623034	0.465725	...	0.562168	0.533517	0.532547	0.368732	0.462188
std	1.184893	1.126869	1.166408	1.128527	1.177127	...	1.013791	1.096624	1.109566	1.054264	1.083575
min	-2.250894	-2.093878	-2.333184	-2.706717	-2.692306	...	-1.883003	-2.635207	-3.261826	-3.022999	-2.256594
25%	-0.349649	-0.310156	-0.250327	-0.180374	-0.378999	...	-0.278949	-0.188104	-0.178580	-0.253884	-0.301883
50%	0.549731	0.526950	0.490512	0.683688	0.476582	...	0.542980	0.489923	0.667405	0.333904	0.474960
75%	1.317116	1.282290	1.452413	1.357684	1.336189	...	1.302980	1.265066	1.296486	1.078843	1.250051
max	3.297516	3.401776	3.945229	3.513855	3.359325	...	2.875418	3.643230	3.286674	2.773621	2.894728

Figure 2.2: Statistics of Dataset 2.

Exploration Data Analysis

Correlation Plot

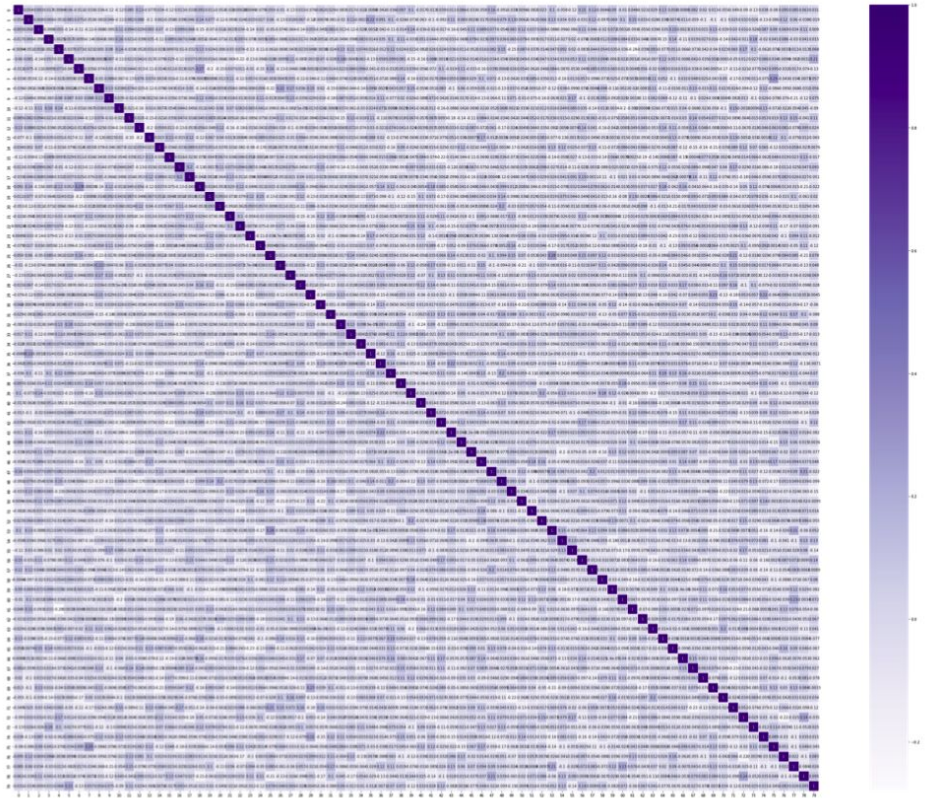


Figure 2.3: Correlation matrix of features.

There are not any special correlations between features

Exploration Data Analysis

Time Series Decomposition

80 features

	0	1	2	3	4	...	75	76	77	78	79
0	1.504400	-1.752114	1.845582	2.356077	2.082137	...	1.321498	0.092595	-0.307086	1.980530	2.759271
1	-0.611505	-0.374634	0.532804	0.904019	0.713504	...	0.697199	-1.628836	-0.425414	-0.751046	-0.164086
2	0.206142	1.819827	1.371899	0.855946	0.036729	...	-0.576464	2.903690	1.368655	0.982679	2.417441
3	-1.442224	0.105535	0.493595	0.518771	1.062785	...	1.194212	0.168850	-0.273234	-0.865923	-0.581641
4	-1.250162	-1.222312	-1.076287	-1.332252	0.946180	...	1.011281	-1.191288	1.342311	1.091403	1.730275
...	...	...	...	...	...	...	...	...	...	...	...
195	0.594266	1.387261	1.566027	1.236675	0.916292	...	0.597874	0.911599	0.272978	2.129287	-0.590149
196	0.614757	0.872680	-2.019999	1.165225	-0.540442	...	0.486028	-2.635207	0.647919	-0.964973	0.483947
197	-1.204028	-0.814811	-0.173549	-0.733251	-1.393954	...	0.453850	0.591489	-0.466551	0.126226	0.582052
198	0.727475	-1.091544	1.591986	2.588876	0.446838	...	-0.424182	0.387061	0.871196	1.014908	0.643886
199	0.161325	0.475174	1.249278	0.720656	-0.925763	...	-1.225119	-0.473057	-0.750419	-0.437075	-0.448496

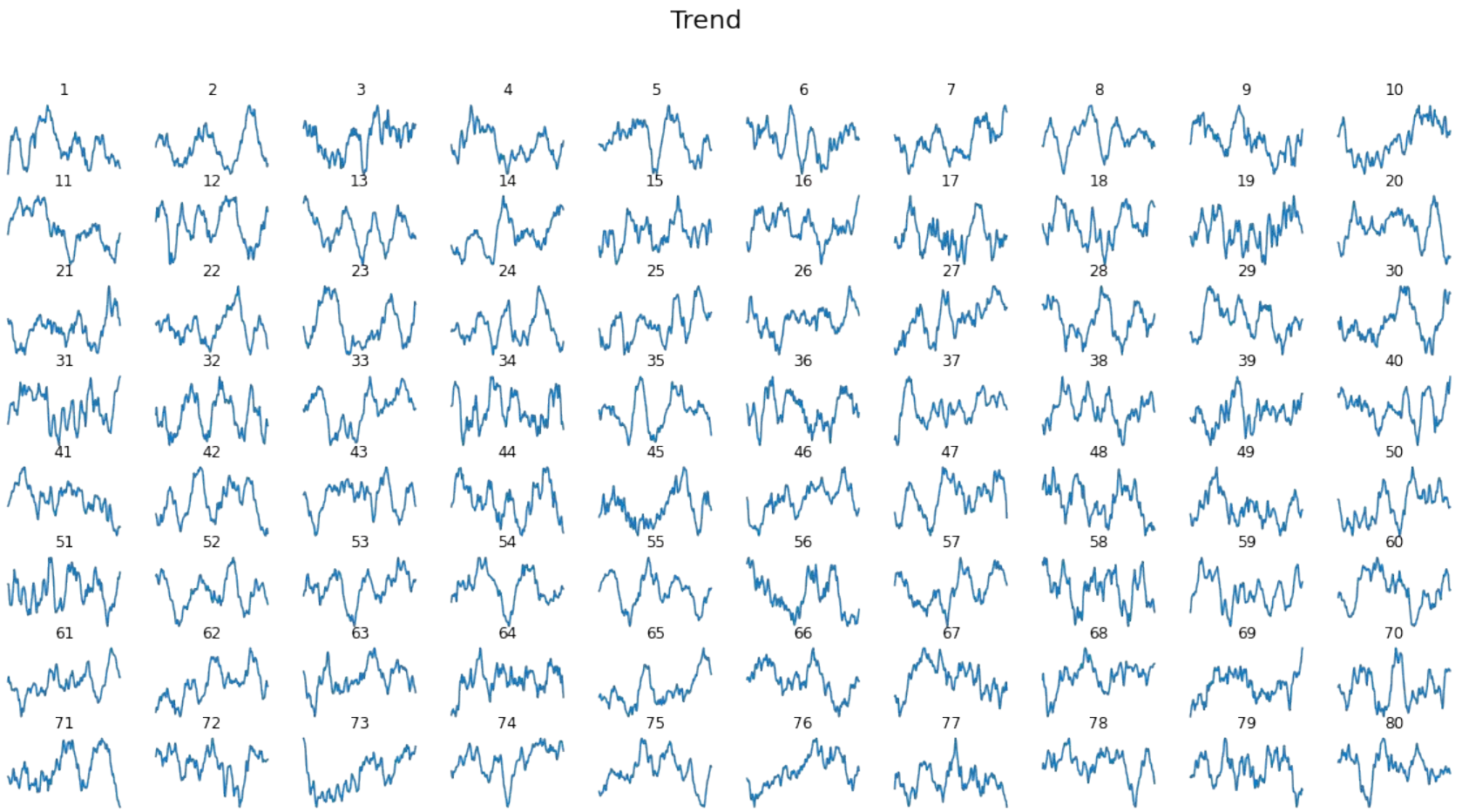
200 rows × 80 columns

200 time-stemp

It may be Tabular Timeseries Data!

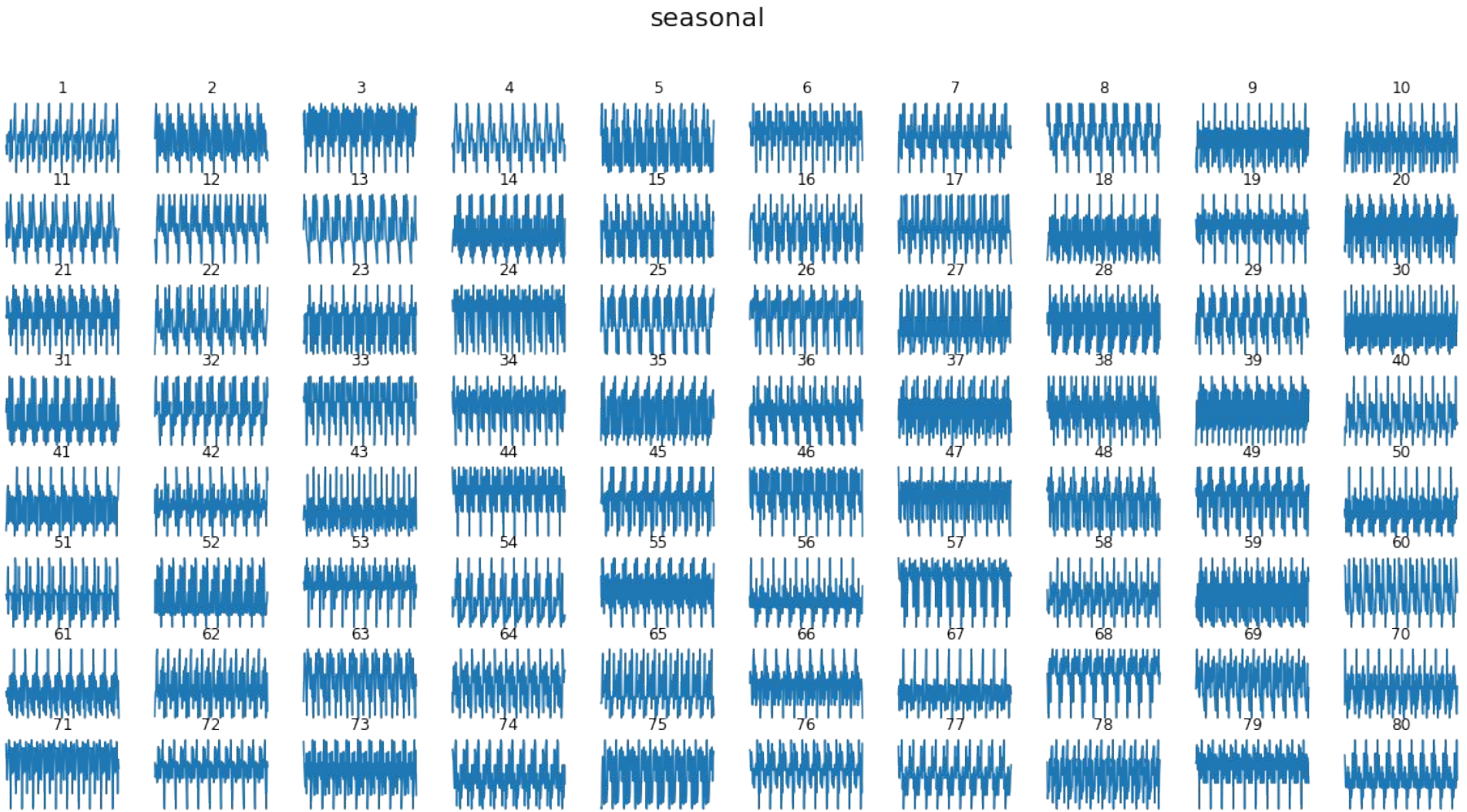
Exploration Data Analysis

Time Series Decomposition - Trend, Seasonal, Residual



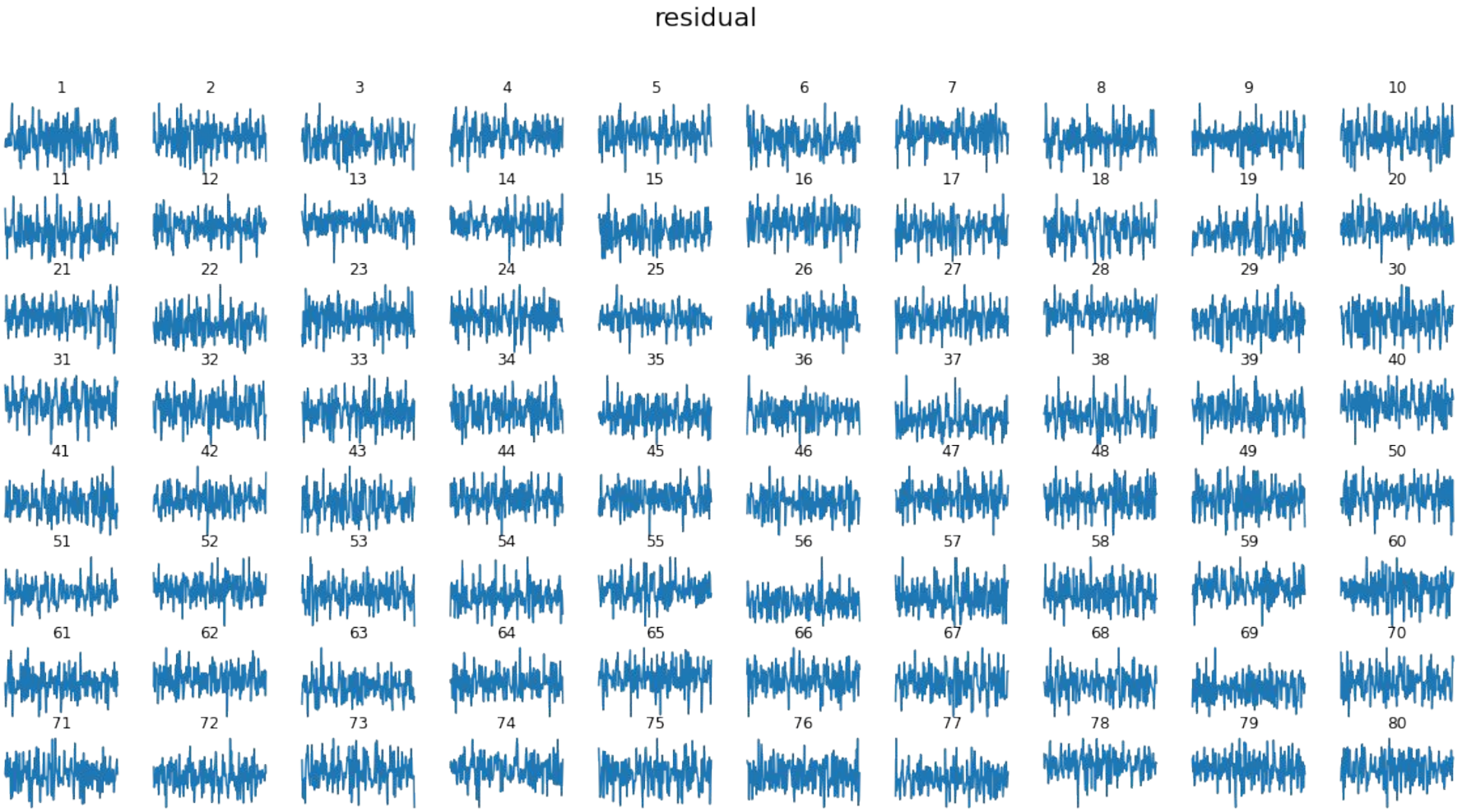
Exploration Data Analysis

Time Series Decomposition - Trend, Seasonal, Residual



Exploration Data Analysis

Time Series Decomposition - Trend, Seasonal, Residual



Exploration Data Analysis

Time Series Decomposition on 1st feature

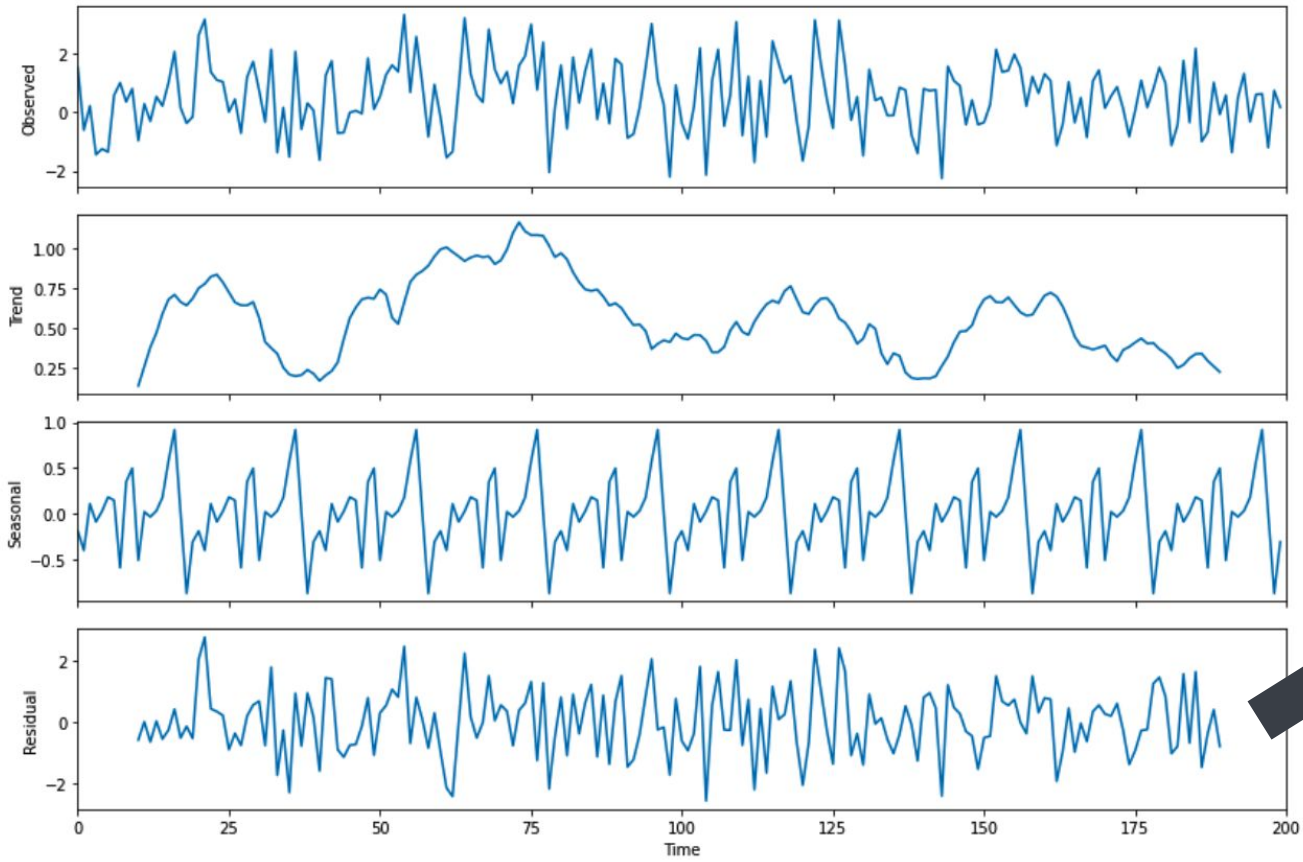
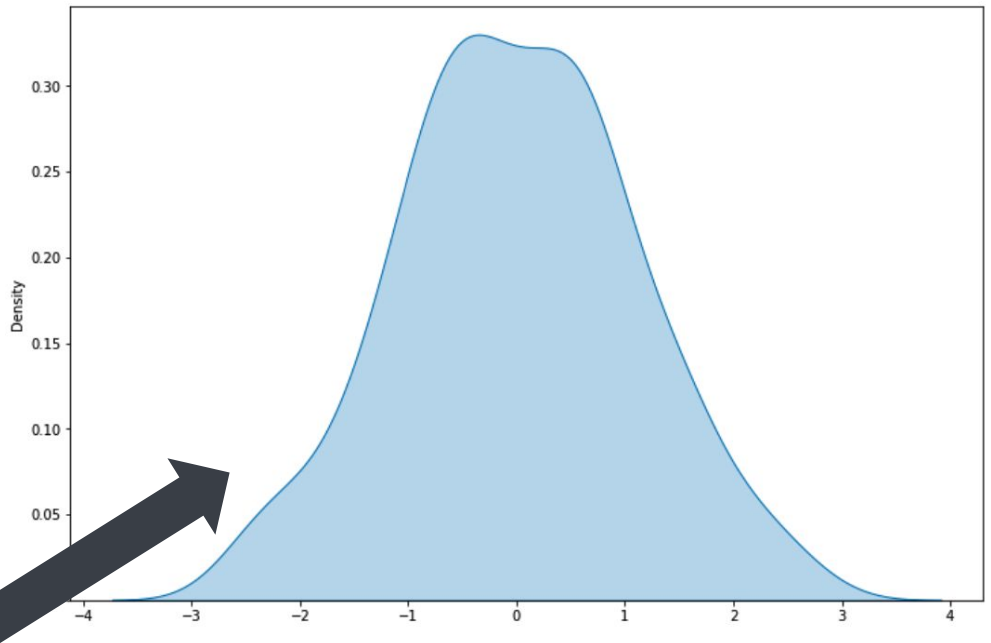


Figure 2.4: Timeseries decomposition of 1st feature.

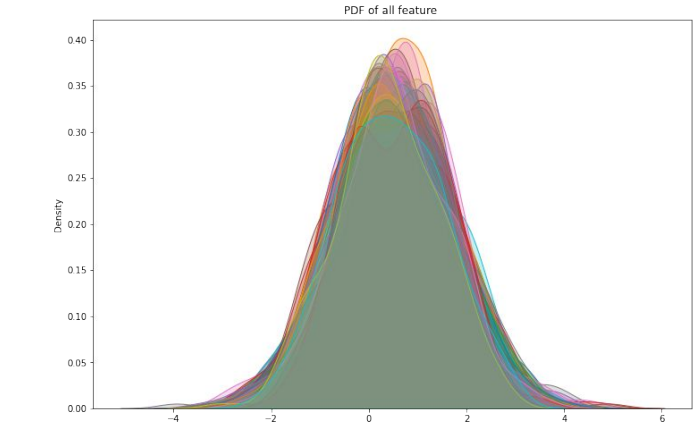
PDF of Residual



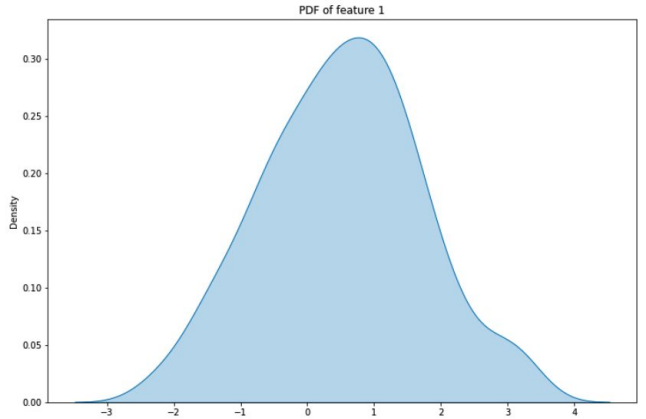
gaussian / no special outliers

Exploration Data Analysis

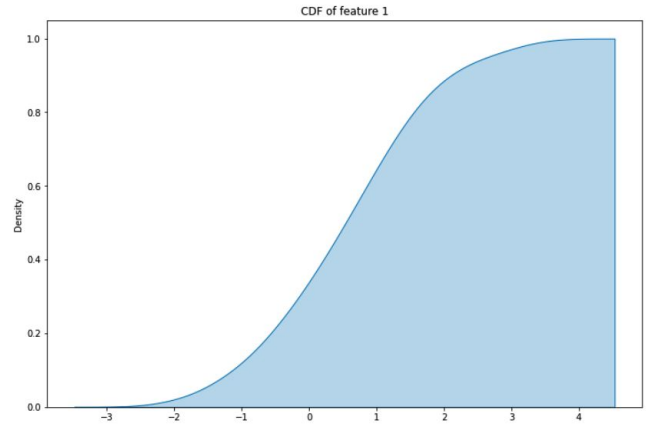
PDF & CDF of all 80 features; Arbitrary Gaussian?



Feature 1



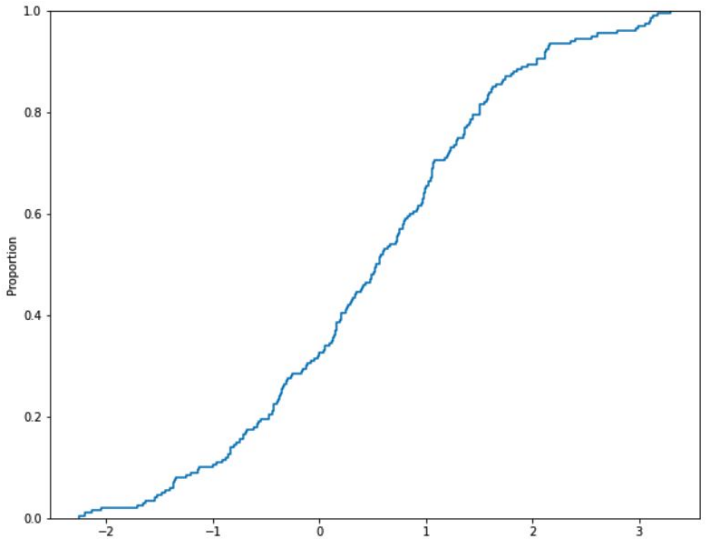
(a) P.D.F of feature 1.



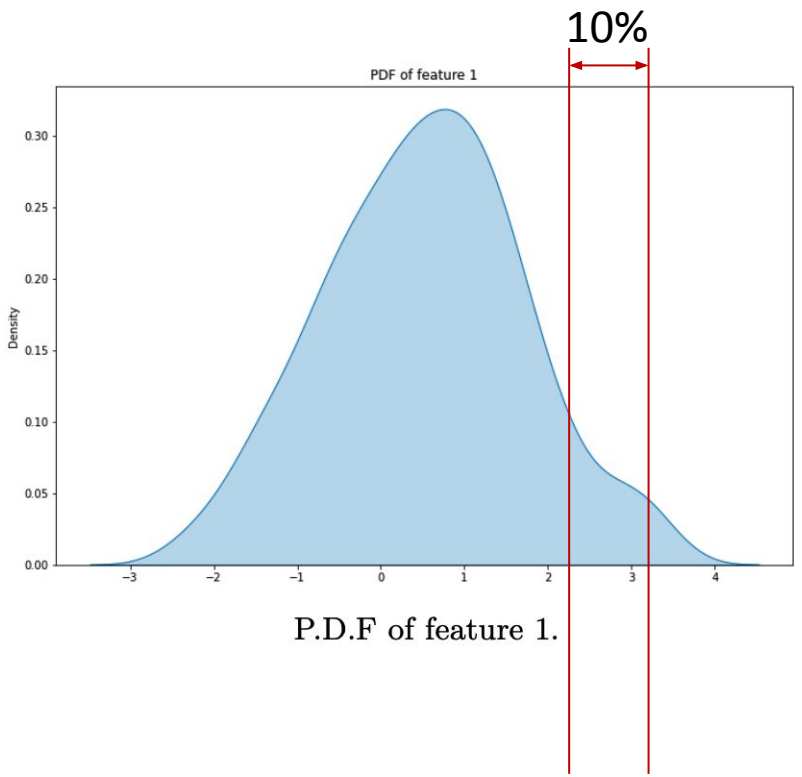
(b) C.D.F of feature 1.

Preprocessing

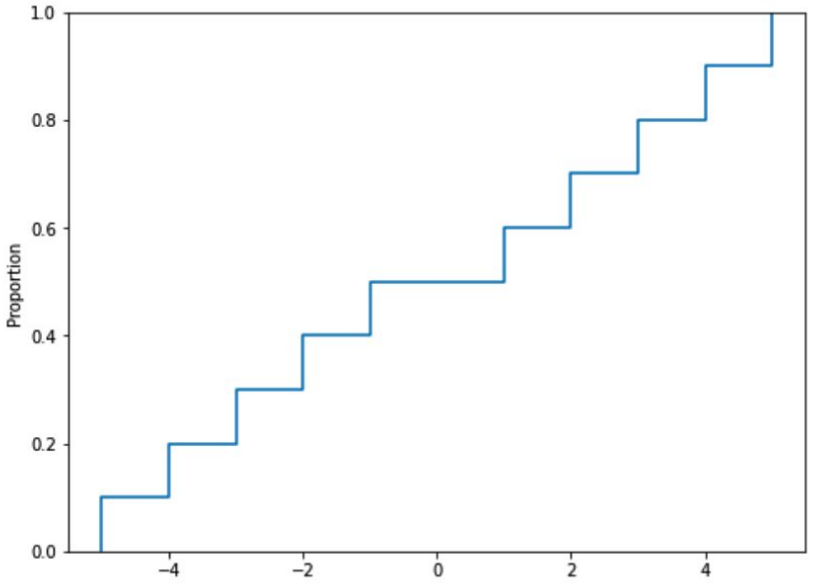
Feature Quantization & Rescaling



(a) C.D.F of feature 1 before pre-processing



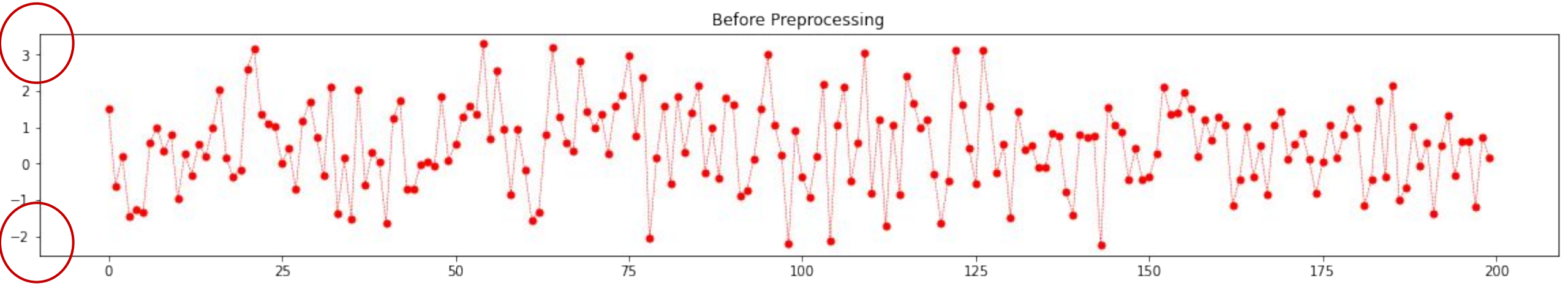
P.D.F of feature 1.

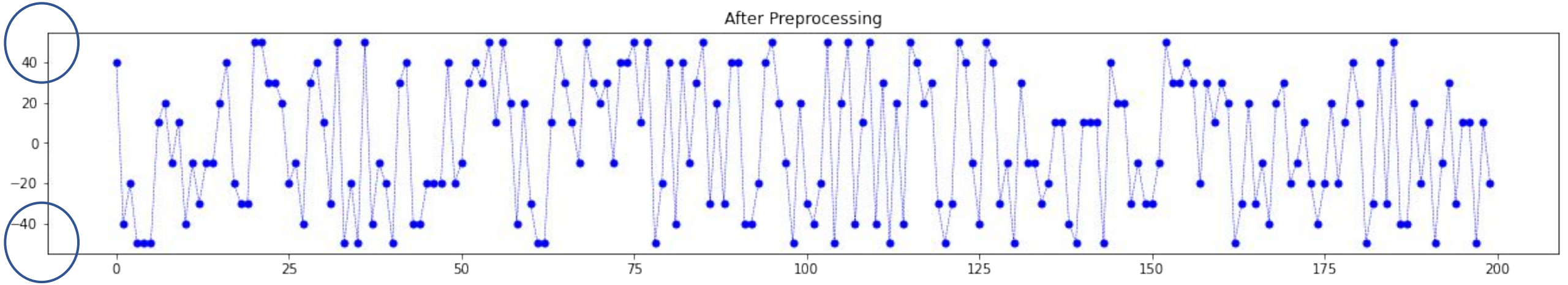


(b) C.D.F of feature 1 after pre-processing.

Preprocessing

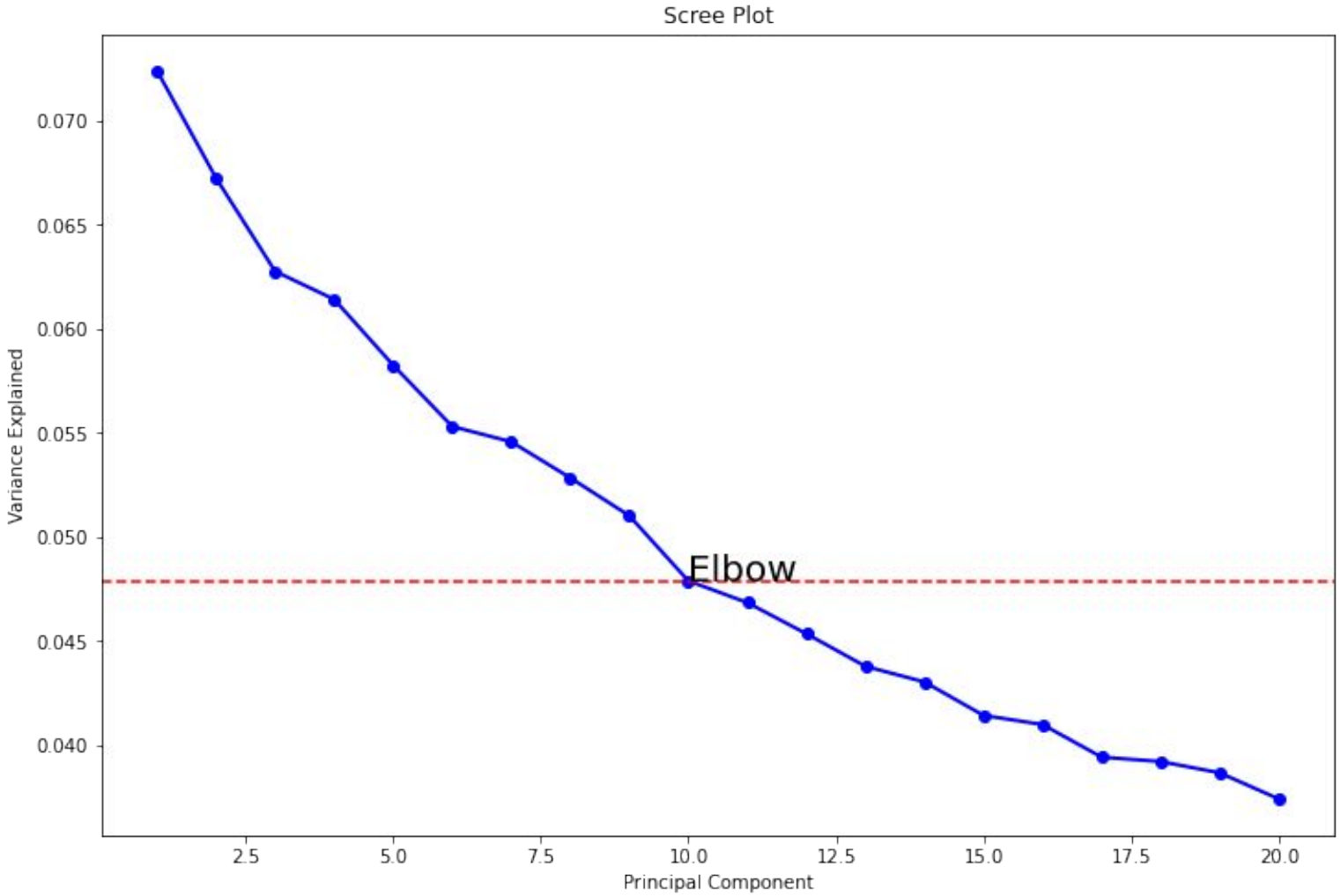
Feature Quantization & Rescaling





Modeling

Principle Component Analysis



Model input

10 PCs
+
preprocessed data

Kernel SVM with
polynomial kernel

Result

Achieve 82.0% accuracy with test data

Follow parameters of NuSVC gives the best performance :

```
{nu = 0.64,
gamma = 'scale',
coef0 = 0.075,
kernel = 'poly',
degree = 3}.
```

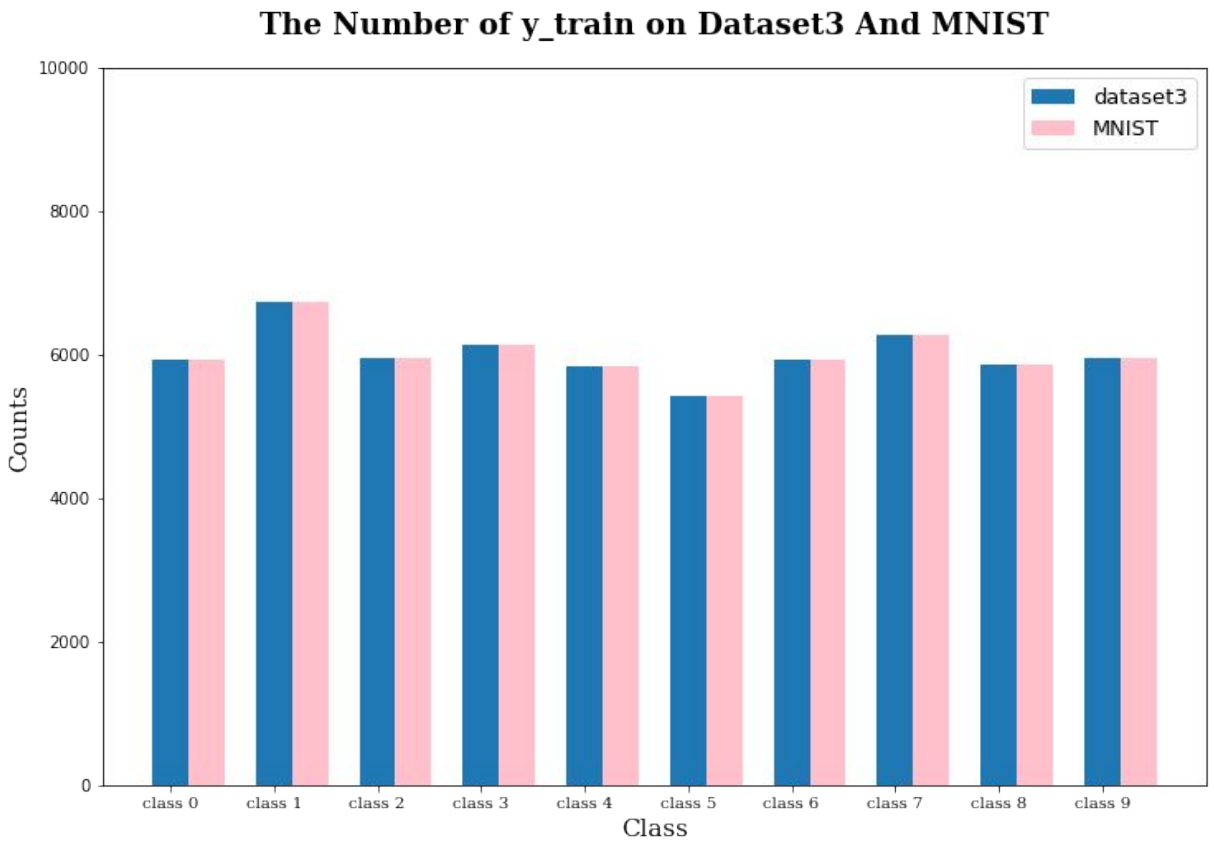
	Precision	Recall	F1-score	Support
0.0	79.09	87.00	82.86	200
1.0	85.56	77.00	81.03	200
Accuracy			**82.00**	400
Macro avg	82.32	82.00	81.96	400
Weighted avg	82.32	82.00	81.96	400

Table 2.1: Overall performance of Dataset2.

Dataset 3

Where is it from?

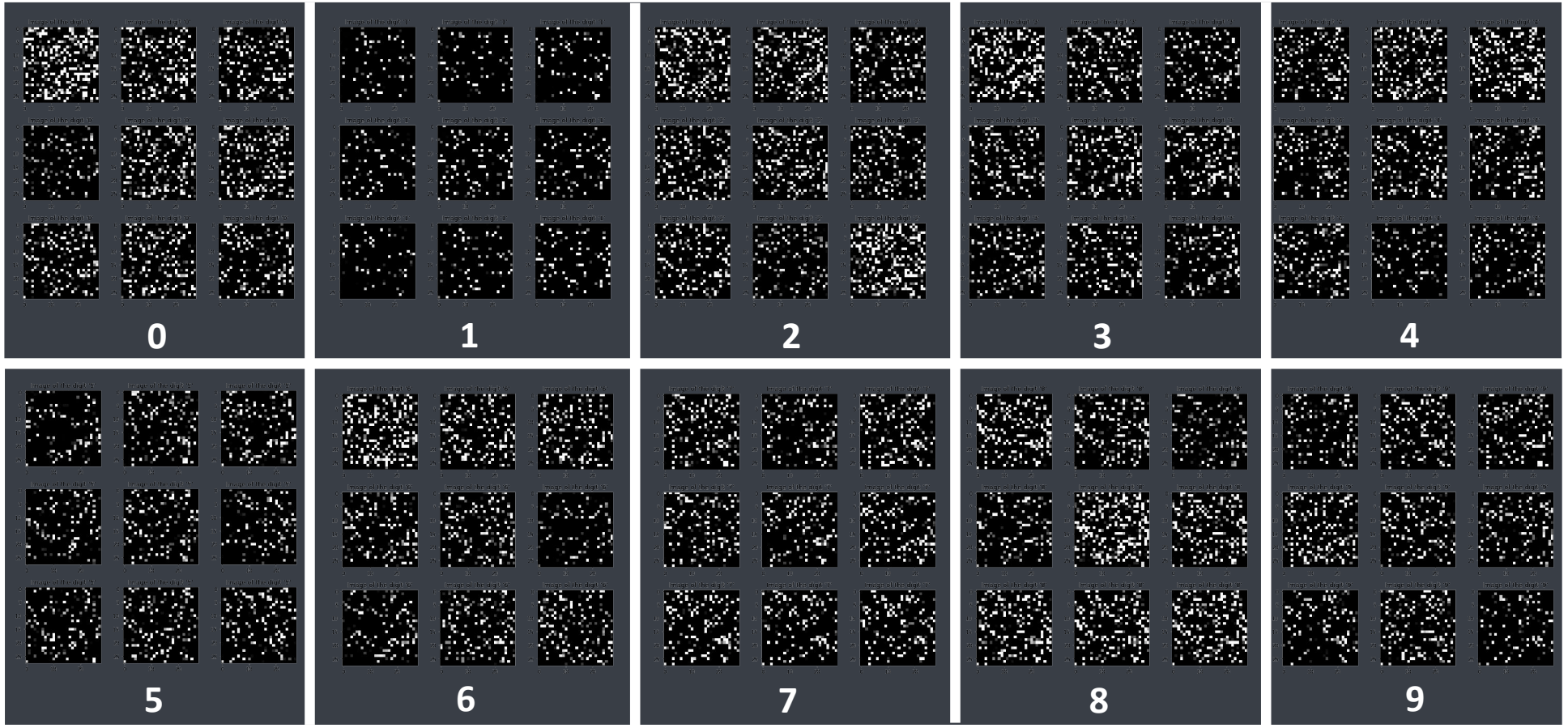
Dataset 3 vs MNIST



- To prove that the given data was derived from MNIST, we compare the MNIST with the given data.
- Figure shows the total occurrences of labels per class in both dataset3 and MNIST. (the number of y_{train})
- We find that the number of dataset 3 labels for training totally corresponds with that of MNIST.

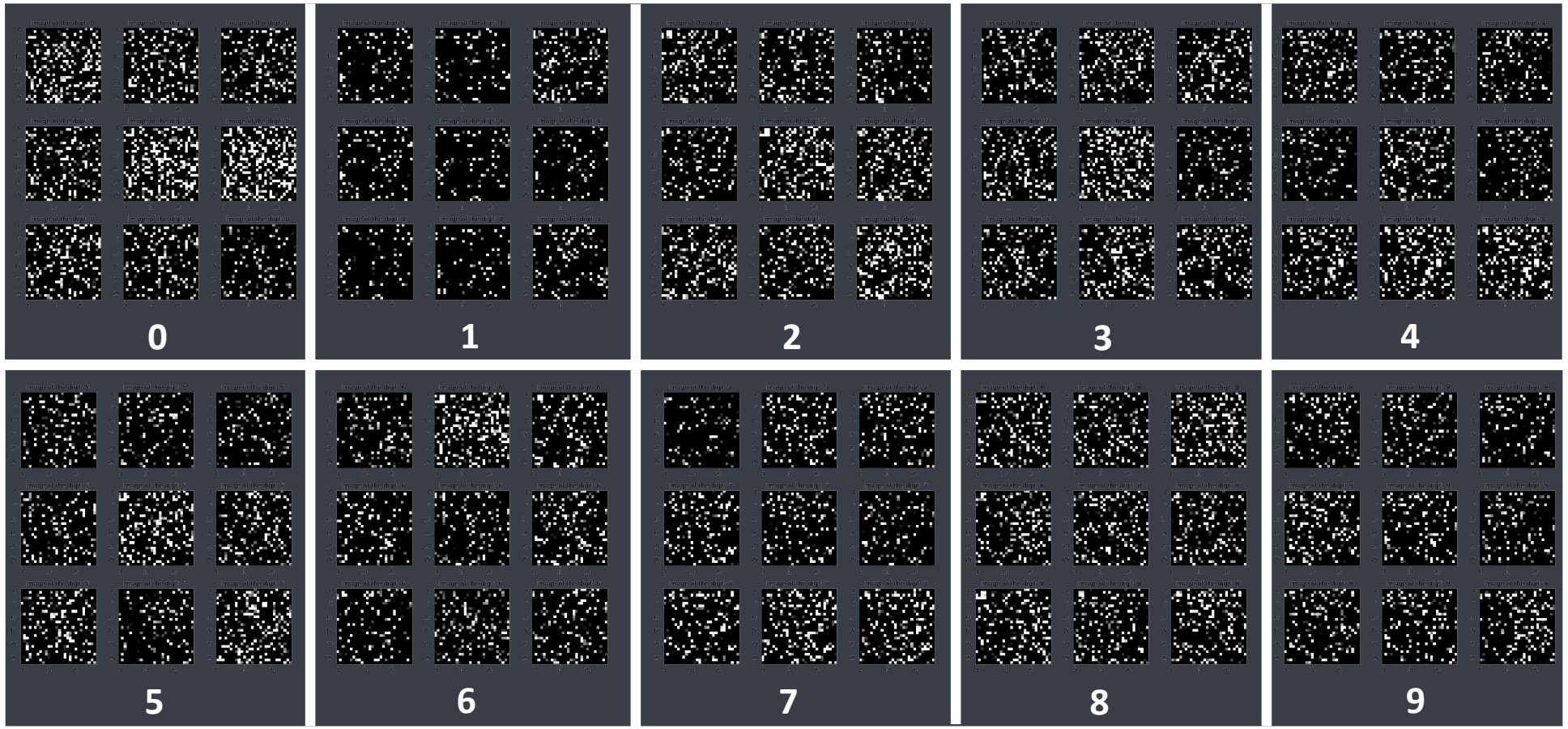
Dataset 3

- Our assumption is that dataset 3 was transformed by permutation invariance on MNIST, which is a different random permutation of the input pixels.



Permuted MNIST

- To achieve that assumption, we try to convert MNIST into Permuted MNIST under permutation invariance method.



Model Architecture & Dataset

```
Train: 192000
Valid: 48000
Test: 60000
FullyConnectedClassifier(
  (layers): Sequential(
    (0): Linear(in_features=784, out_features=500, bias=True)
    (1): LeakyReLU(negative_slope=0.01)
    (2): BatchNorm1d(500, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Linear(in_features=500, out_features=450, bias=True)
    (4): LeakyReLU(negative_slope=0.01)
    (5): BatchNorm1d(450, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): Linear(in_features=450, out_features=400, bias=True)
    (7): LeakyReLU(negative_slope=0.01)
    (8): BatchNorm1d(400, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (9): Linear(in_features=400, out_features=350, bias=True)
    (10): LeakyReLU(negative_slope=0.01)
    (11): BatchNorm1d(350, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (12): Linear(in_features=350, out_features=300, bias=True)
    (13): LeakyReLU(negative_slope=0.01)
    (14): BatchNorm1d(300, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (15): Linear(in_features=300, out_features=200, bias=True)
    (16): LeakyReLU(negative_slope=0.01)
    (17): BatchNorm1d(200, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (18): Linear(in_features=200, out_features=100, bias=True)
    (19): LeakyReLU(negative_slope=0.01)
    (20): BatchNorm1d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (21): Linear(in_features=100, out_features=50, bias=True)
    (22): LeakyReLU(negative_slope=0.01)
    (23): BatchNorm1d(50, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (24): Linear(in_features=50, out_features=10, bias=True)
    (25): LogSoftmax(dim=-1)
  )
)
```

- There is no reason why to use CNN because the dataset has rarely **'spatial information'**.
- We use Fully Connected Neural Network for classification.
- For better performance, we augmented the original dataset 3 under permutation invariance.
  - train: 48,000 □ 192,000
  - valid: 12,000 □ 48,000
  - test: 60,000 □ 60,000
- Adam optimizer (B1 = 0.9 & B2 = 0.999)
- Learning rate: 0.001
- Epoch / Batch_size: 100 / 128

Result

Accuracy 98.37

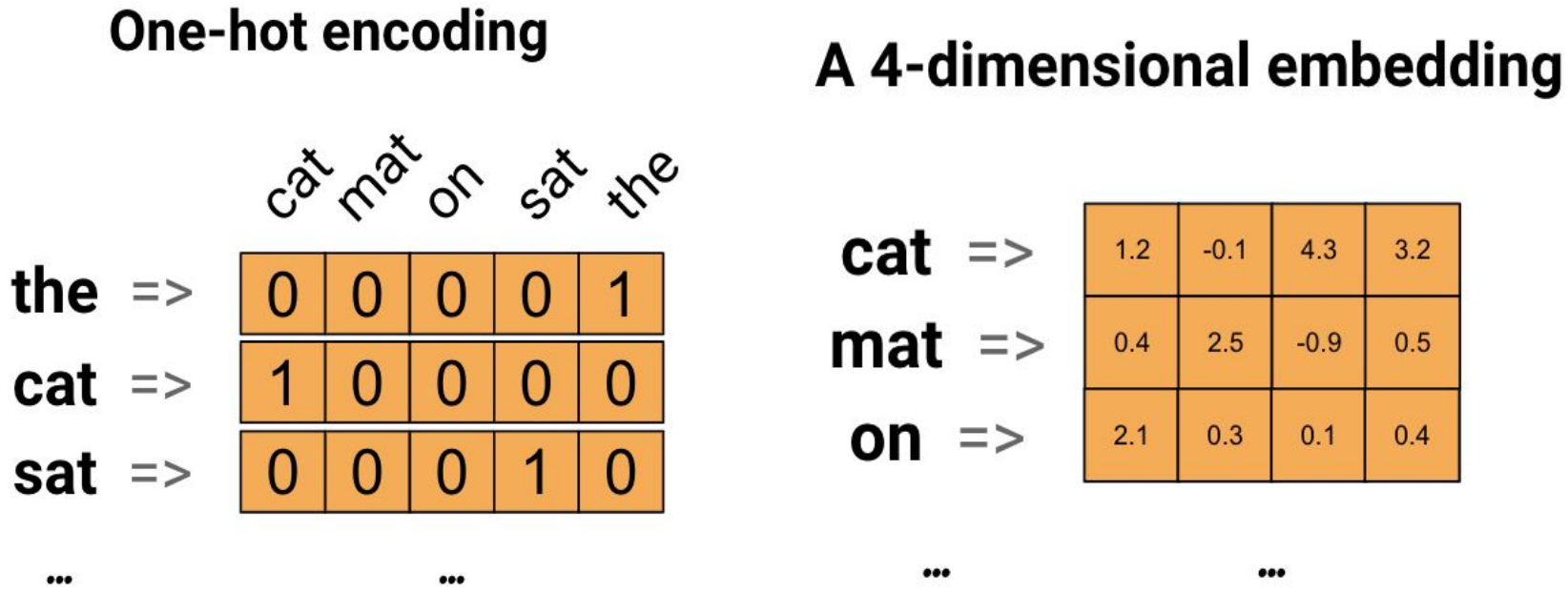
	Precision	Recall	F1-score	Support
0	98.58	98.99	98.78	5952
1	99.06	99.56	99.31	6791
2	97.54	98.87	98.20	6026
3	98.70	97.62	98.16	6084
4	98.50	98.69	98.59	5780
5	98.02	98.02	98.02	5454
6	98.51	98.79	98.65	5957
7	98.72	98.14	98.43	6231
8	97.99	97.47	97.73	5890
9	97.93	97.36	97.65	5835
Accuracy			**98.37**	60000
Macro avg	98.36	98.35	98.35	60000
Weighted avg	98.37	98.37	98.37	60000

Table 3.1: Overall performance of Dataset 3.

Dataset 4

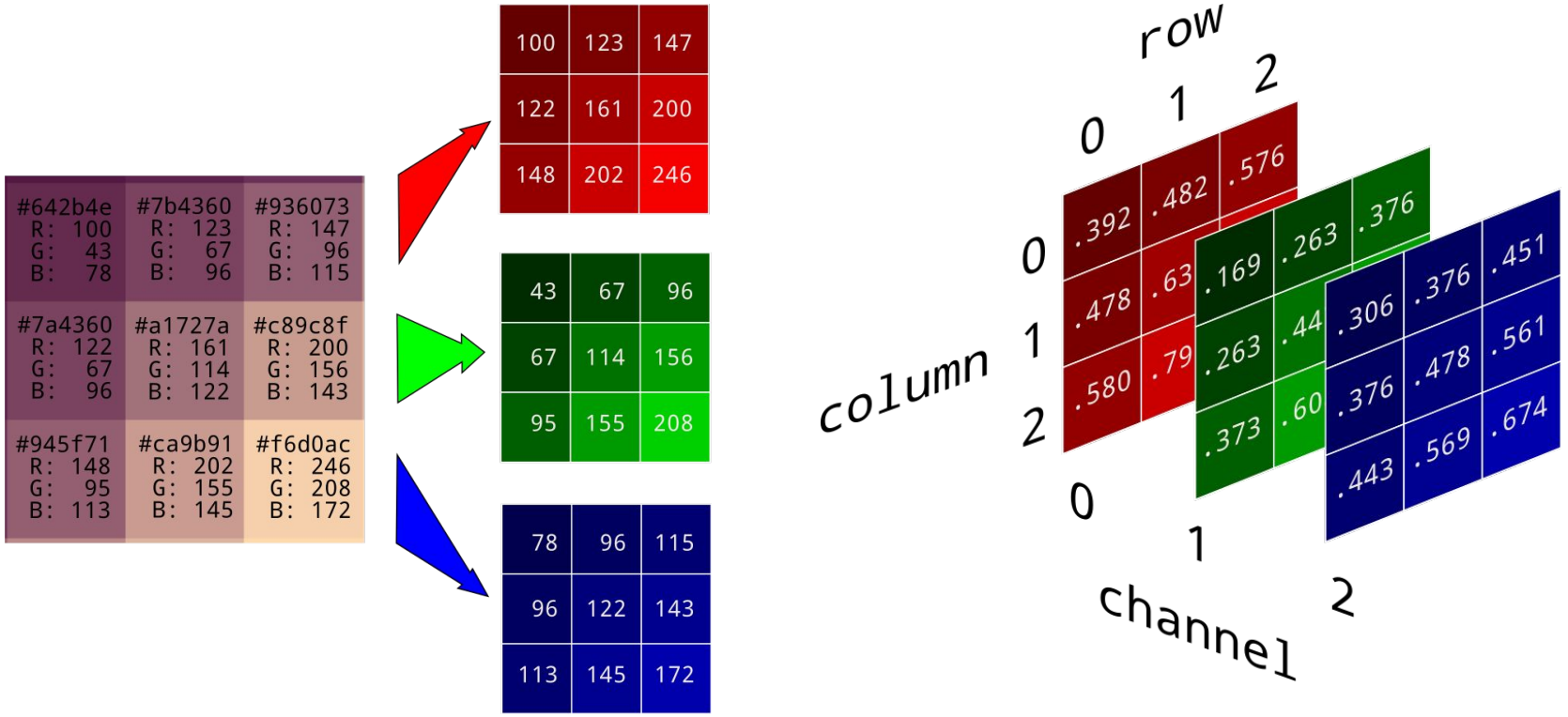
Assumption

- Structured, or Unstructured That is the question.
  - Text data



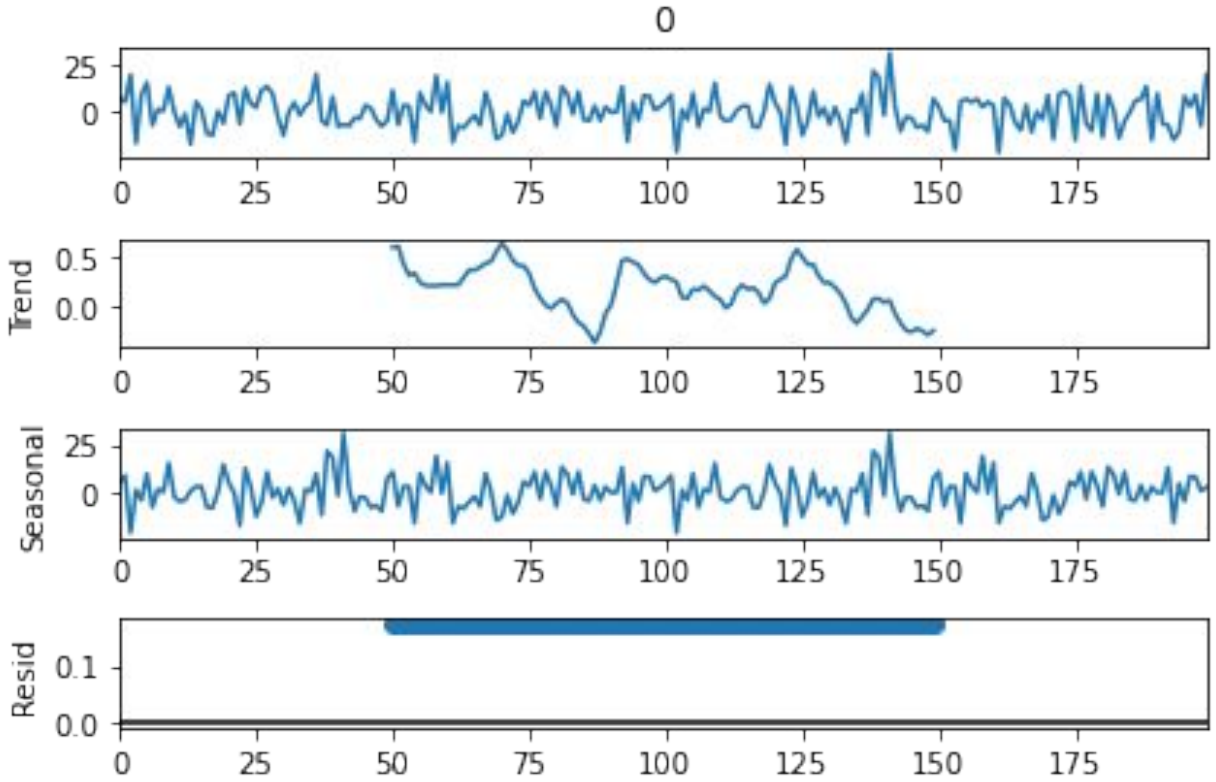
Assumption

- Structured, or Unstructured That is the question.
  - Image data



Assumption

- Structured, or Unstructured That is the question.
  - Time series



Assumption

- Structured, or Unstructured That is the question.
  - Tabular data

86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
499	-10.679374	-9.651669	6.792266	-13.999066	8.633502	6.769784	-0.294137	-3.996679	1.761109	6.174899	14.666311	5.448175	-9.070846	0.0
407	4.083088	5.397036	1.236472	1.947566	-14.723931	2.015775	0.651363	-11.328531	-8.783979	-9.427665	-5.062393	1.146165	-8.858193	0.0
048	13.869284	7.876907	3.593203	6.526867	-1.112752	4.408863	12.838681	-0.015059	-7.224026	4.189209	0.947371	8.556511	8.303979	1.0
678	-21.434269	17.225373	-6.955900	-1.947885	0.131306	-0.142287	5.658618	-5.476792	-11.356855	-3.662806	-1.897227	-15.536735	-12.395929	0.0
920	18.526174	4.677469	-1.205310	1.303600	1.988148	-5.143531	-15.307037	1.150768	7.218858	-28.466678	1.957533	-3.721173	3.586937	1.0

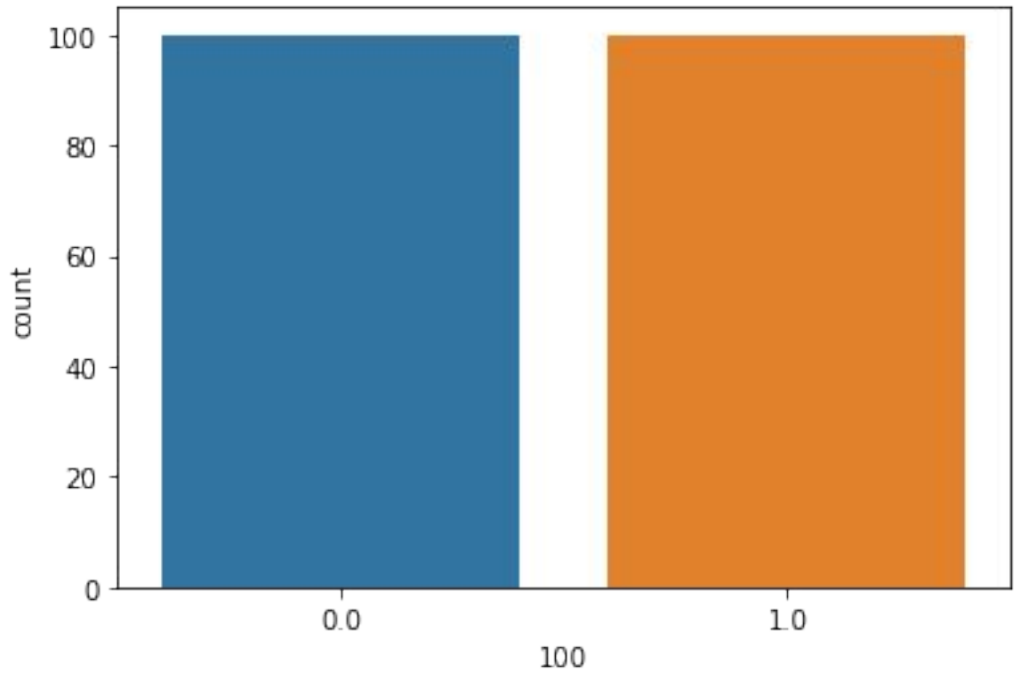
```
1 test.head()
```

86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
527432	9.547160	22.752521	-6.694436	5.770033	5.342211	10.671910	4.629977	2.286529	6.910975	8.381256	-7.081508	-7.208932	4.803683	0.0
310663	-7.081378	-3.593928	-0.657925	-12.417282	4.594153	-2.838235	13.185244	-8.247236	-13.010013	12.369666	8.409057	-1.963031	0.937939	1.0
167924	1.187060	-10.740223	0.586079	-3.288586	7.356371	-28.541481	-12.505706	-5.411032	1.844423	13.609114	3.398033	-0.930907	9.663914	0.0
422845	3.727698	6.853378	9.686923	-3.504703	9.140568	4.454891	2.131894	-1.976306	5.347609	8.621202	10.976914	-0.575334	1.159929	1.0
975283	-14.685234	-7.402291	3.841841	-2.271321	-4.717496	-0.970174	21.874679	14.640316	1.648271	-4.654442	20.922828	-2.311537	8.694663	0.0

First five rows of train and test data

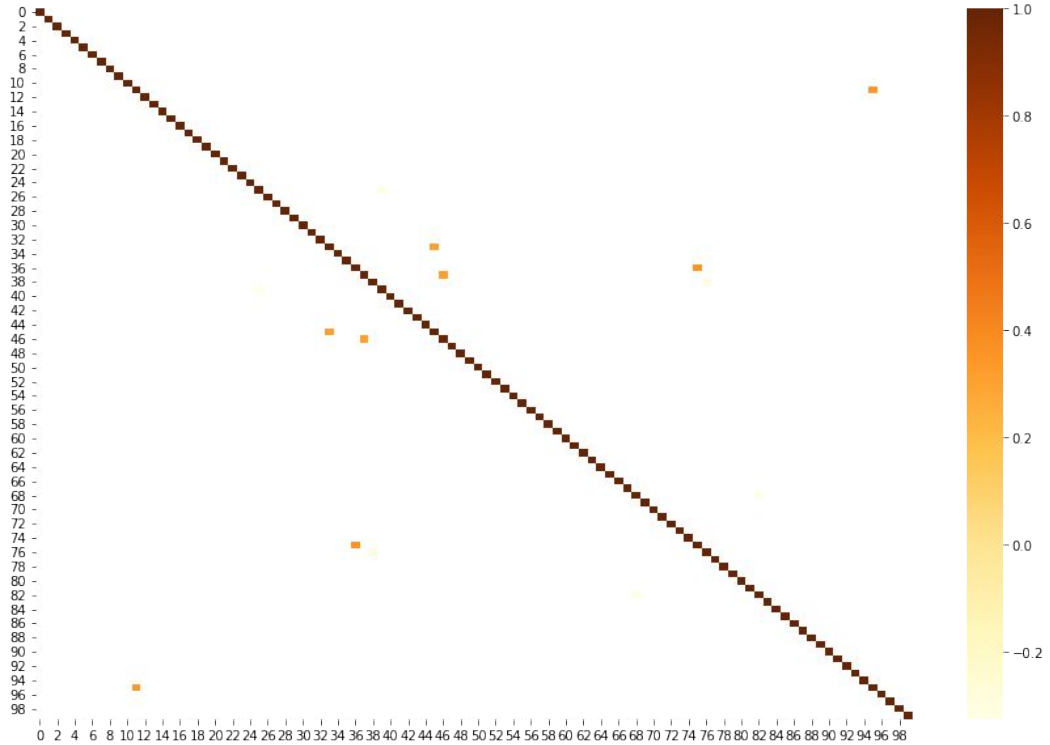
Modeling

- Look before you build your model
  - Class label imbalance?



Modeling

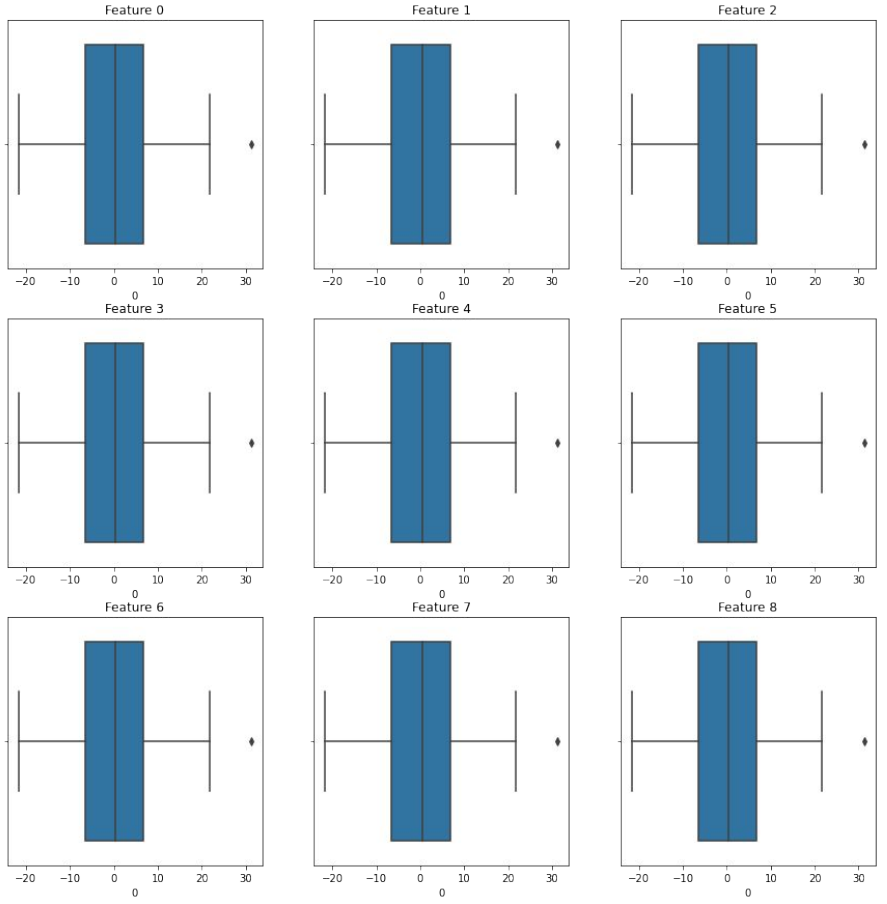
- Look before you build your model
  - Highly Correlated Features?



Orange dots represent
“abs(corr)>= 0.3”

Modeling

- Look before you build your model
  - Extreme Outlier?



Modeling

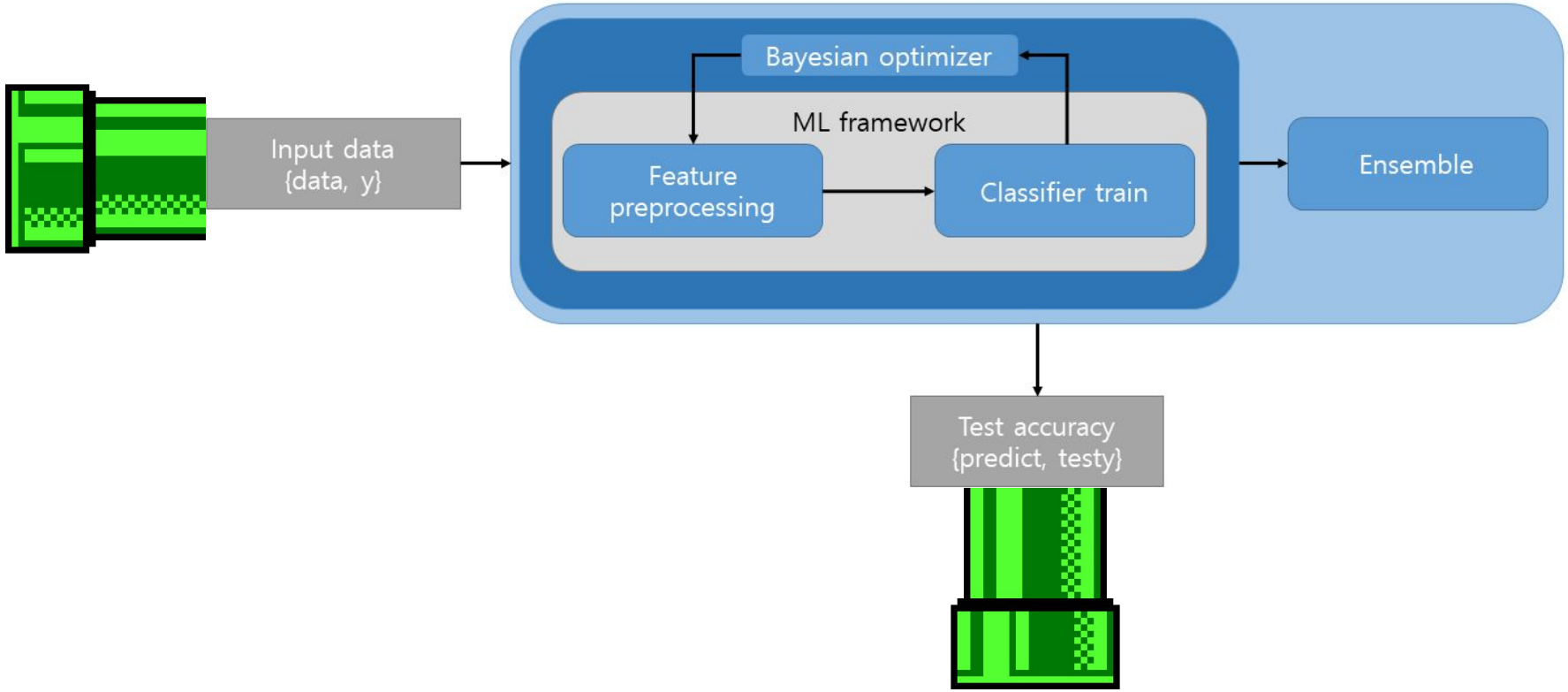
- Look before you build your model
  - Missing values?

	feature	null_ratio
0	0	0.0
1	1	0.0
2	2	0.0
3	3	0.0
4	4	0.0
...	...	...
96	96	0.0
97	97	0.0
98	98	0.0
99	99	0.0
100	100	0.0

101 rows × 2 columns

Modeling

- Look before you build your model
  - Pipeline !



Result

- Don't count your chickens before they hatch
  - LDA to QDA

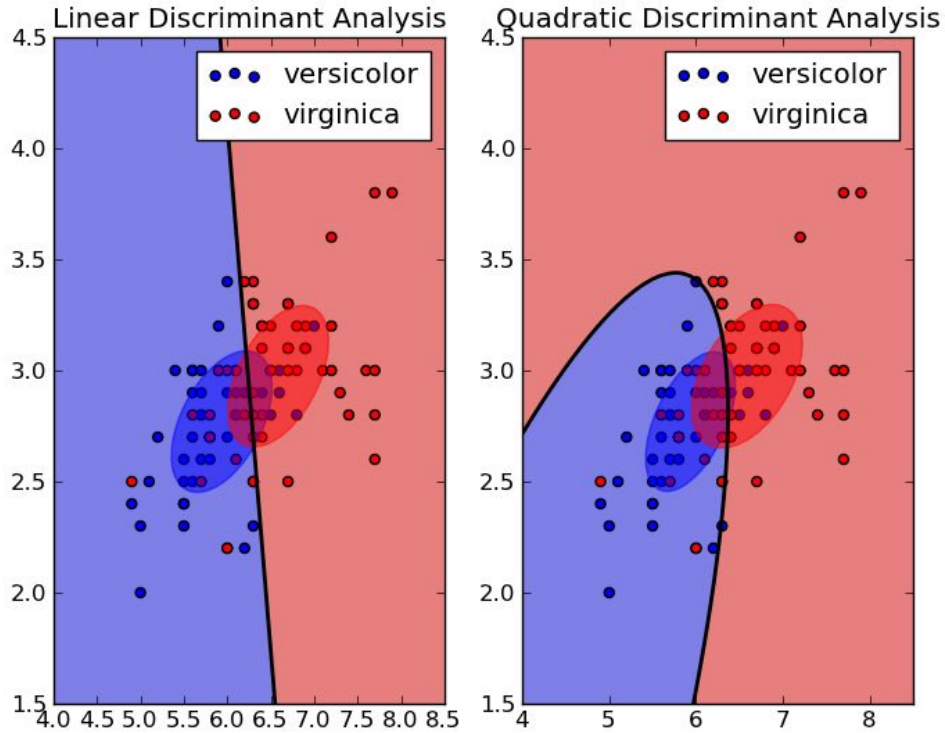
model_id	rank	ensemble_weight	type	cost	duration	config_id	train_loss	seed
215	63	0.04	lda	0.0	0.565253	214	0.000000	0
361	28	0.04	lda	0.0	0.549816	360	0.000000	0
164	42	0.03	lda	0.0	0.662376	163	0.000000	0
141	37	0.03	lda	0.0	54.763352	140	0.000000	0
293	5	0.03	lda	0.0	0.657380	292	0.000000	0
213	65	0.03	lda	0.0	0.582082	212	0.000000	0
319	15	0.03	lda	0.0	0.535382	318	0.000000	0
311	13	0.03	lda	0.0	0.544596	310	0.000000	0
248	55	0.02	lda	0.0	0.568721	247	0.000000	0

model_id	rank	ensemble_weight	type	cost	duration	config_id	train_loss	seed
306	63	0.04	qda	0.0	2.140178	305	0.0	0
443	28	0.04	qda	0.0	1.601399	442	0.0	0
271	42	0.03	qda	0.0	2.083739	270	0.0	0
257	37	0.03	qda	0.0	2.936898	256	0.0	0
380	5	0.03	qda	0.0	2.793200	379	0.0	0
302	65	0.03	qda	0.0	2.073994	301	0.0	0
411	15	0.03	qda	0.0	1.526780	410	0.0	0
407	13	0.03	qda	0.0	1.500182	406	0.0	0
329	55	0.02	qda	0.0	1.646588	328	0.0	0

LDA & QDA based ensemble model

Result

- Don't count your chickens before they hatch
  - Flexible Decision Boundary



Result

- Don't count your chickens before they hatch
  - Bang! Bullseye!

	Precision	Recall	F1-score	Support
0.0	100	100	100	100
1.0	100	100	100	100
Accuracy			100	200
Marco avg	100	100	100	200
Weight avg	100	100	100	200

Table 4.1: Overall performance of Dataset4.

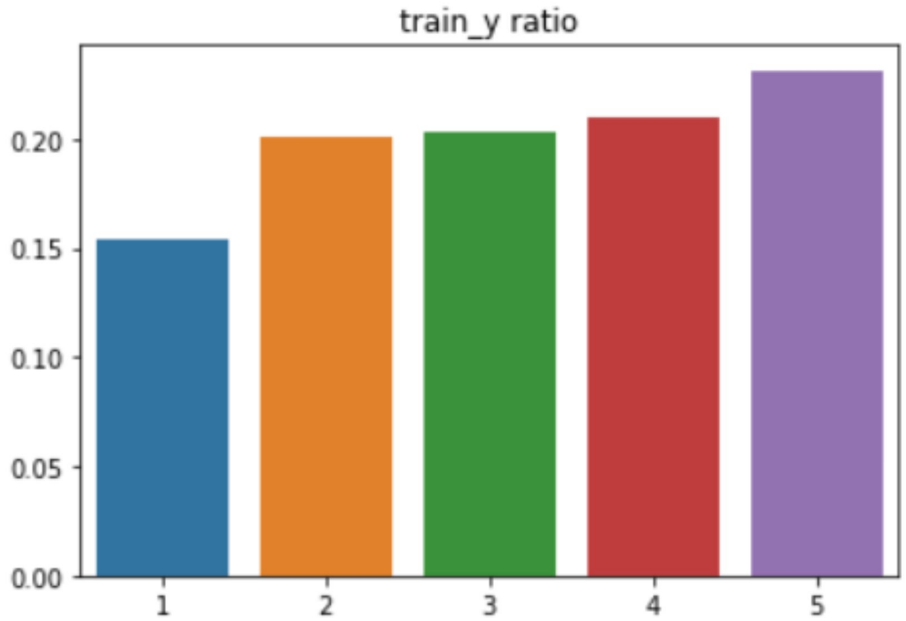
Dataset 5

Data Description

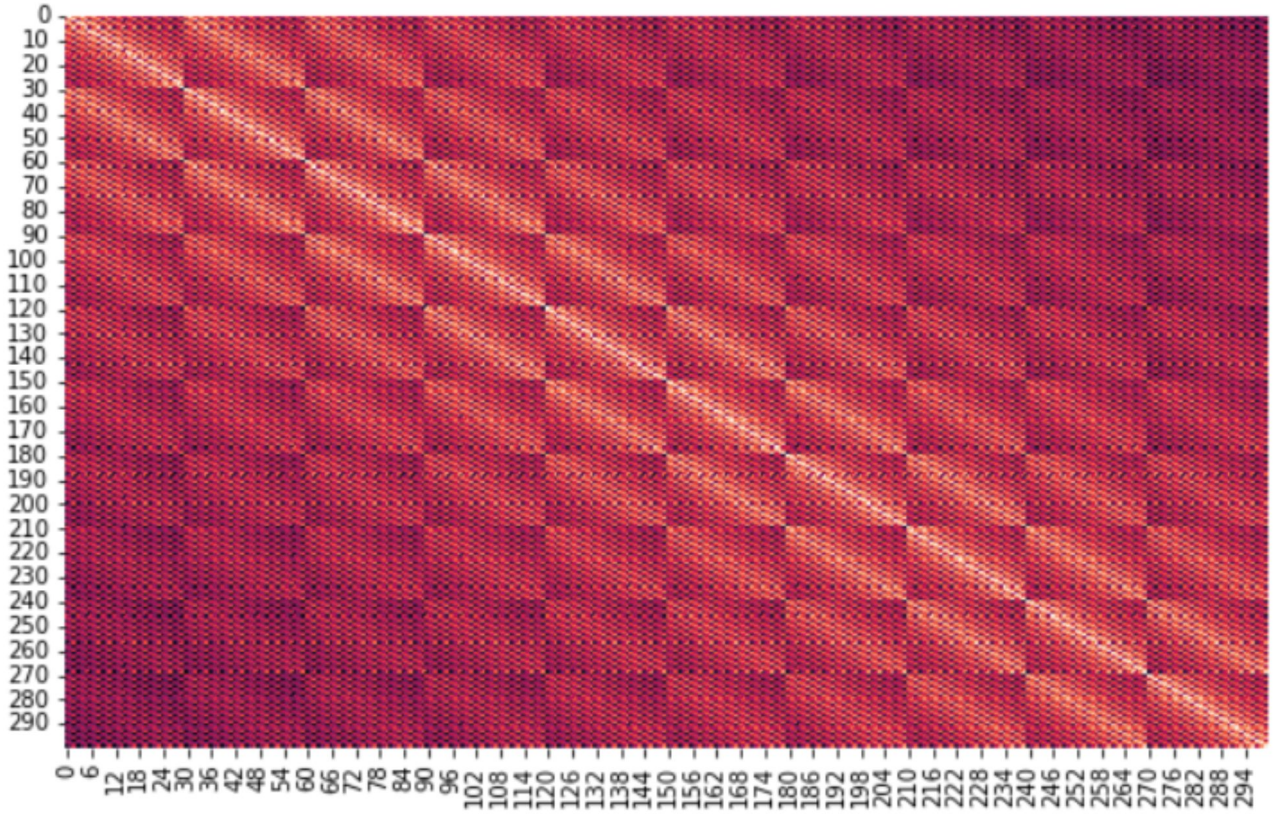
- Data shape

```
train x : (391500, 300) train y : (391500,)
test x : (167650, 300) test y : (167650,)
```

- Label Ratio



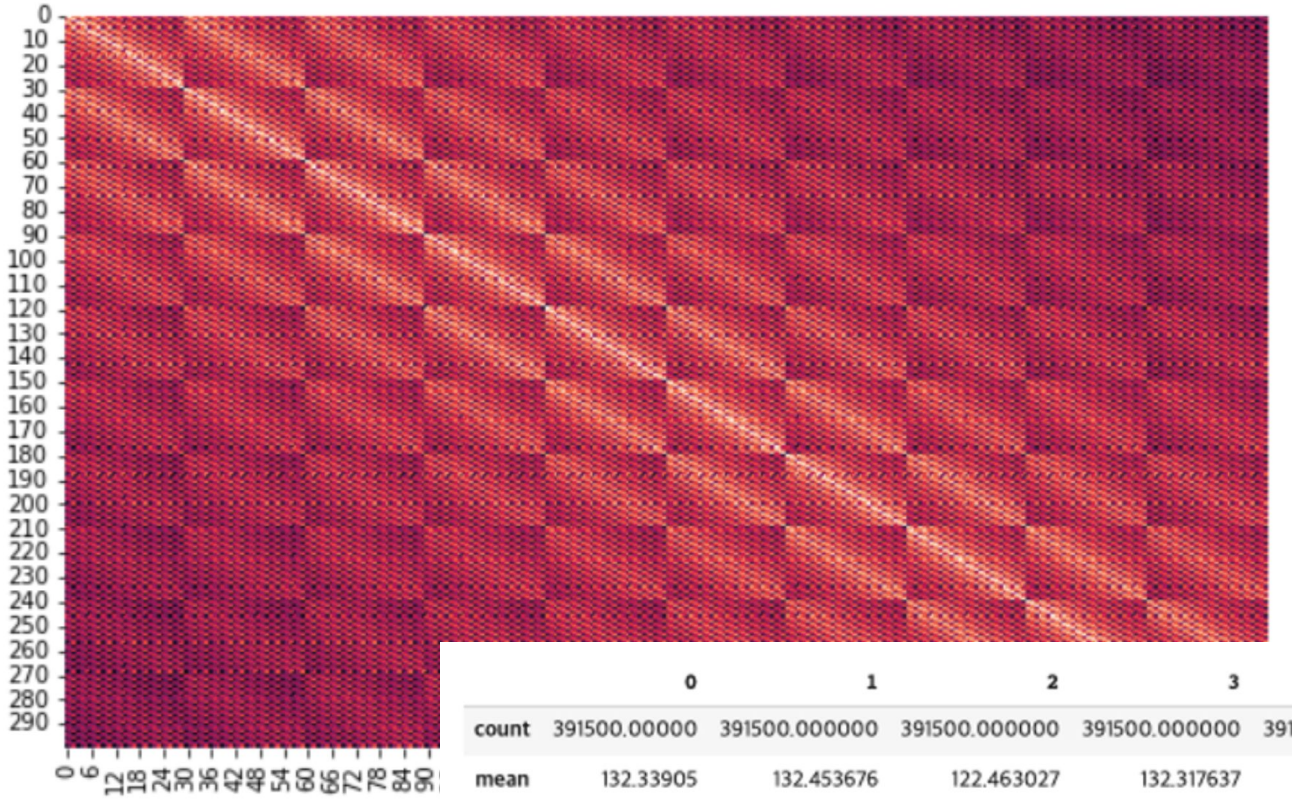
What is this data?



					4	..	296	297	298	299
					500.000000	...	391500.000000	391500.000000	391500.000000	391500.000000
					132.440342	...	121.241226	131.494613	131.486598	121.199269
std	65.72883	66.009981	73.766281	65.703052	65.993691	...	73.454690	65.609769	65.843740	73.441240
min	0.00000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000
25%	81.00000	80.000000	59.000000	81.000000	80.000000	...	59.000000	80.000000	80.000000	59.000000
50%	133.00000	131.000000	116.000000	133.000000	131.000000	...	115.000000	132.000000	130.000000	115.000000
75%	184.00000	185.000000	184.000000	184.000000	185.000000	...	182.000000	183.000000	184.000000	182.000000
max	255.00000	255.000000	255.000000	255.000000	255.000000	...	255.000000	255.000000	255.000000	255.000000

8 rows × 300 columns

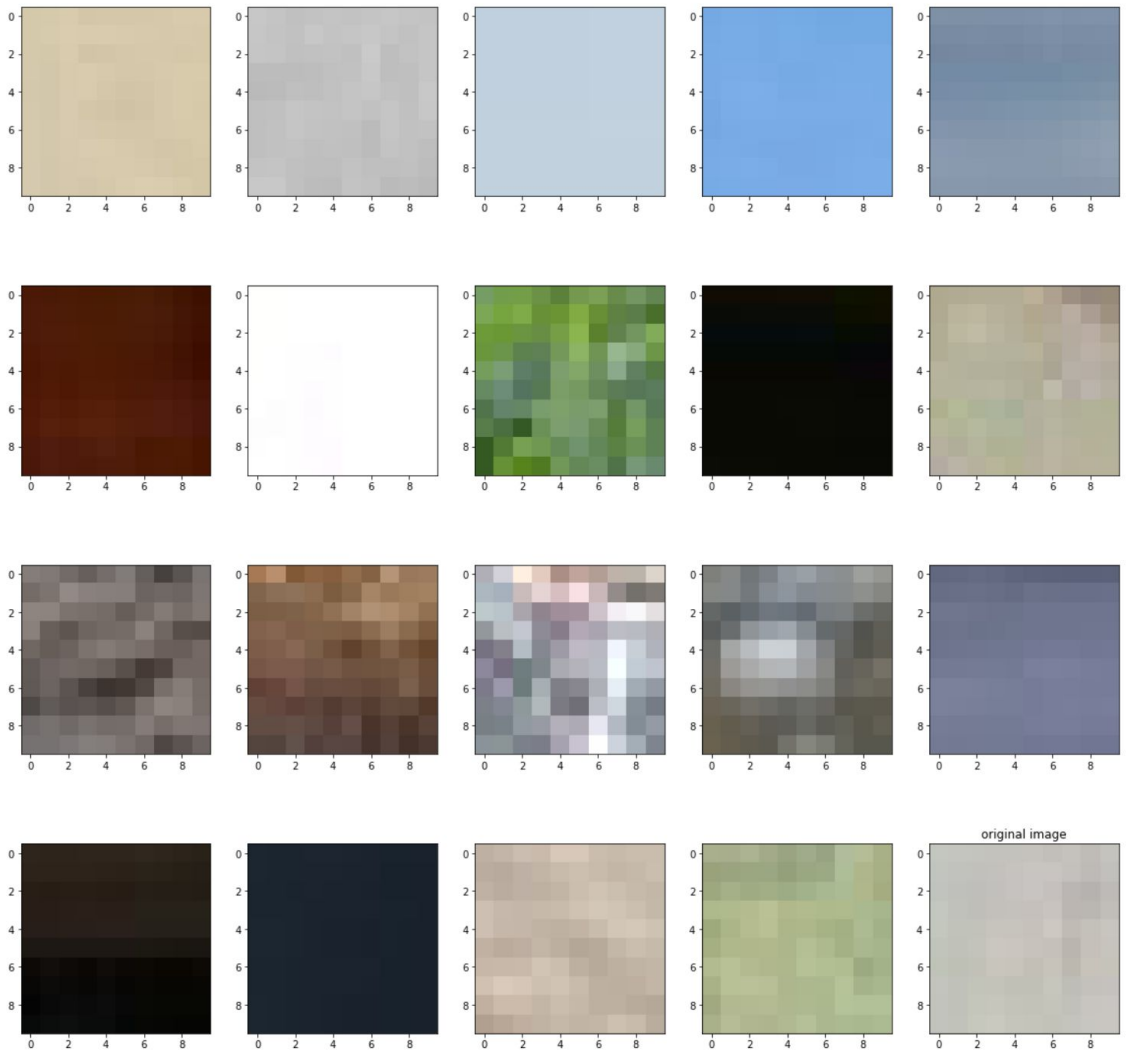
What is this data?



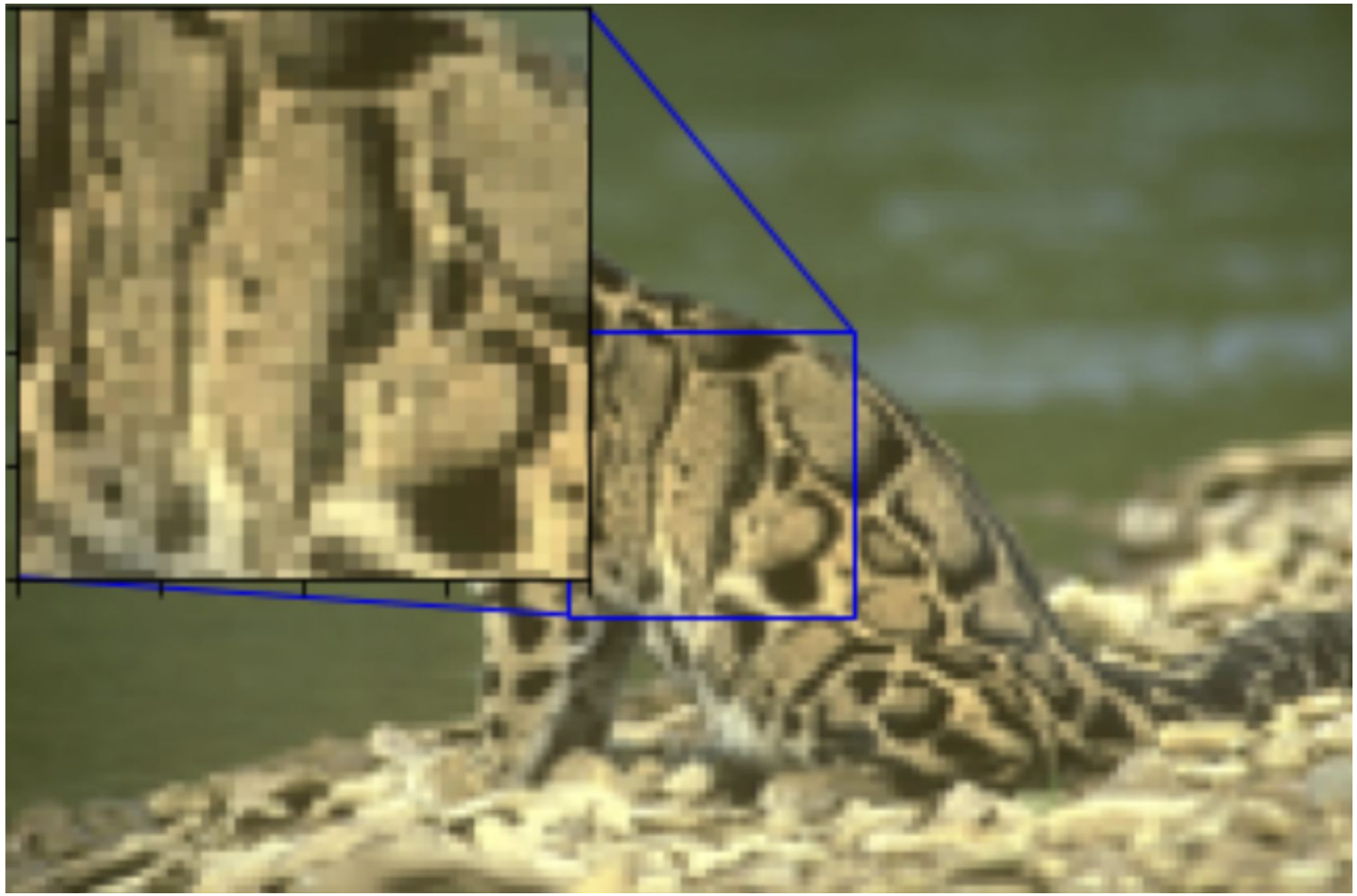
	0	1	2	3	4	..	296	297	298	299
count	391500.00000	391500.000000	391500.000000	391500.000000	391500.000000	...	391500.000000	391500.000000	391500.000000	391500.000000
mean	132.33905	132.453676	122.463027	132.317637	132.440342	...	121.241226	131.494613	131.486598	121.199269
std	65.72883	66.009981	73.766281	65.703052	65.993691	...	73.454690	65.609769	65.843740	73.441240
min	0.00000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000
25%	81.00000	80.000000	59.000000	81.000000	80.000000	...	59.000000	80.000000	80.000000	59.000000
50%	133.00000	131.000000	116.000000	133.000000	131.000000	...	115.000000	132.000000	130.000000	115.000000
75%	184.00000	185.000000	184.000000	184.000000	185.000000	...	182.000000	183.000000	184.000000	182.000000
max	255.00000	255.000000	255.000000	255.000000	255.000000	...	255.000000	255.000000	255.000000	255.000000

8 rows × 300 columns

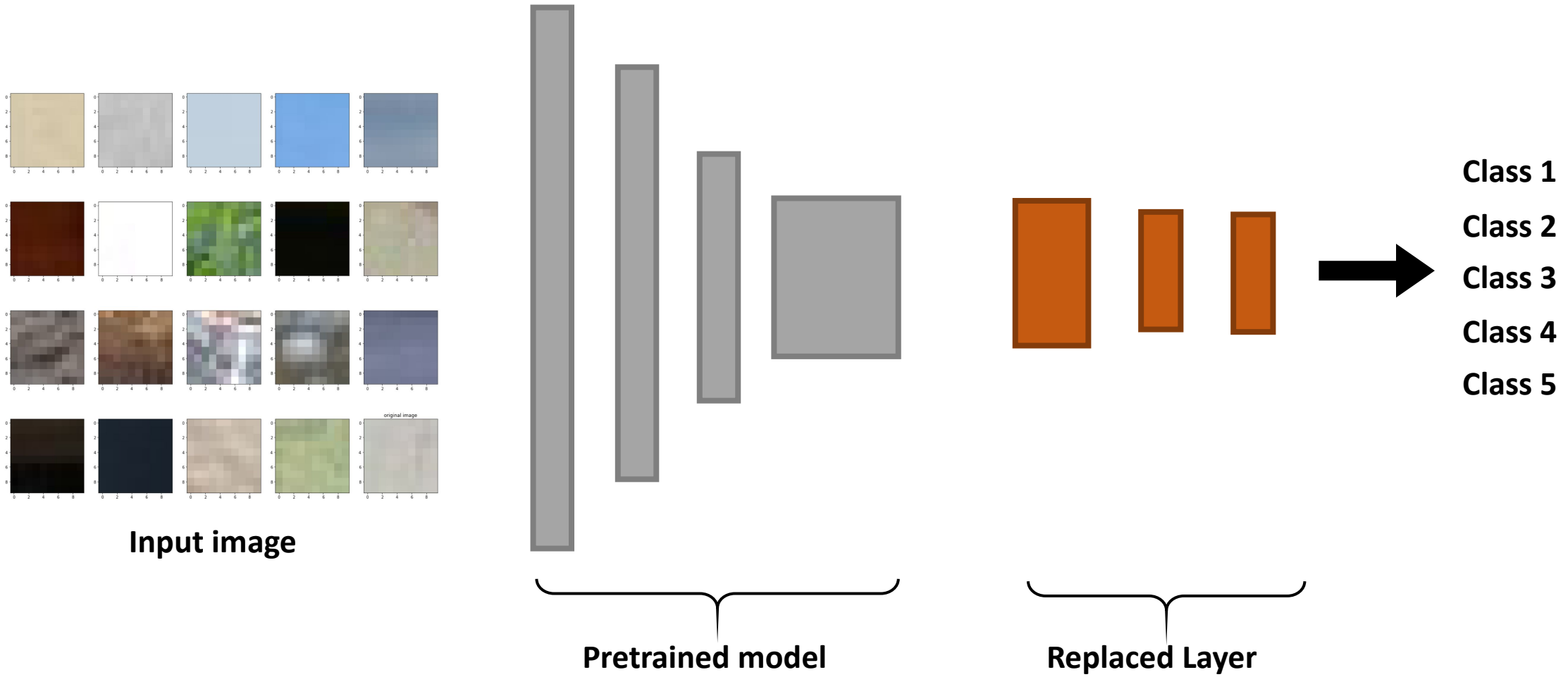
Image?



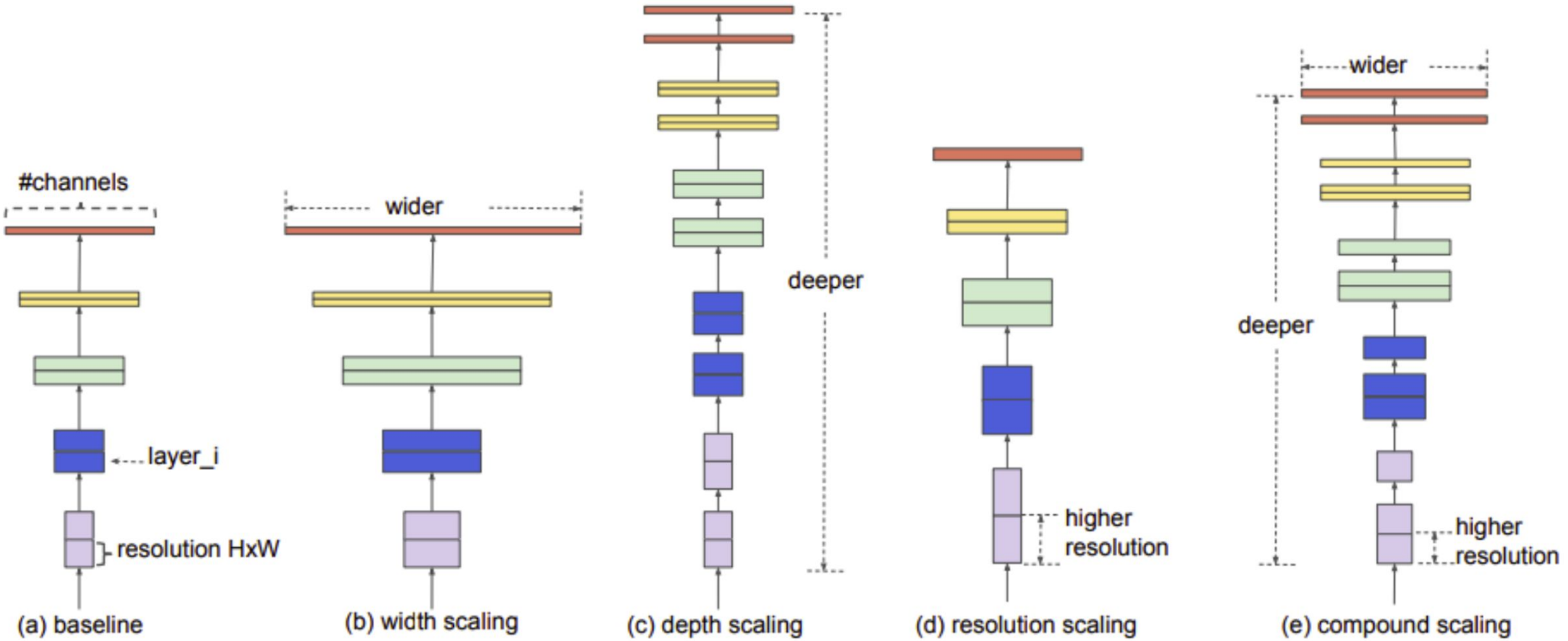
Image?



Train with a well pre-trained model



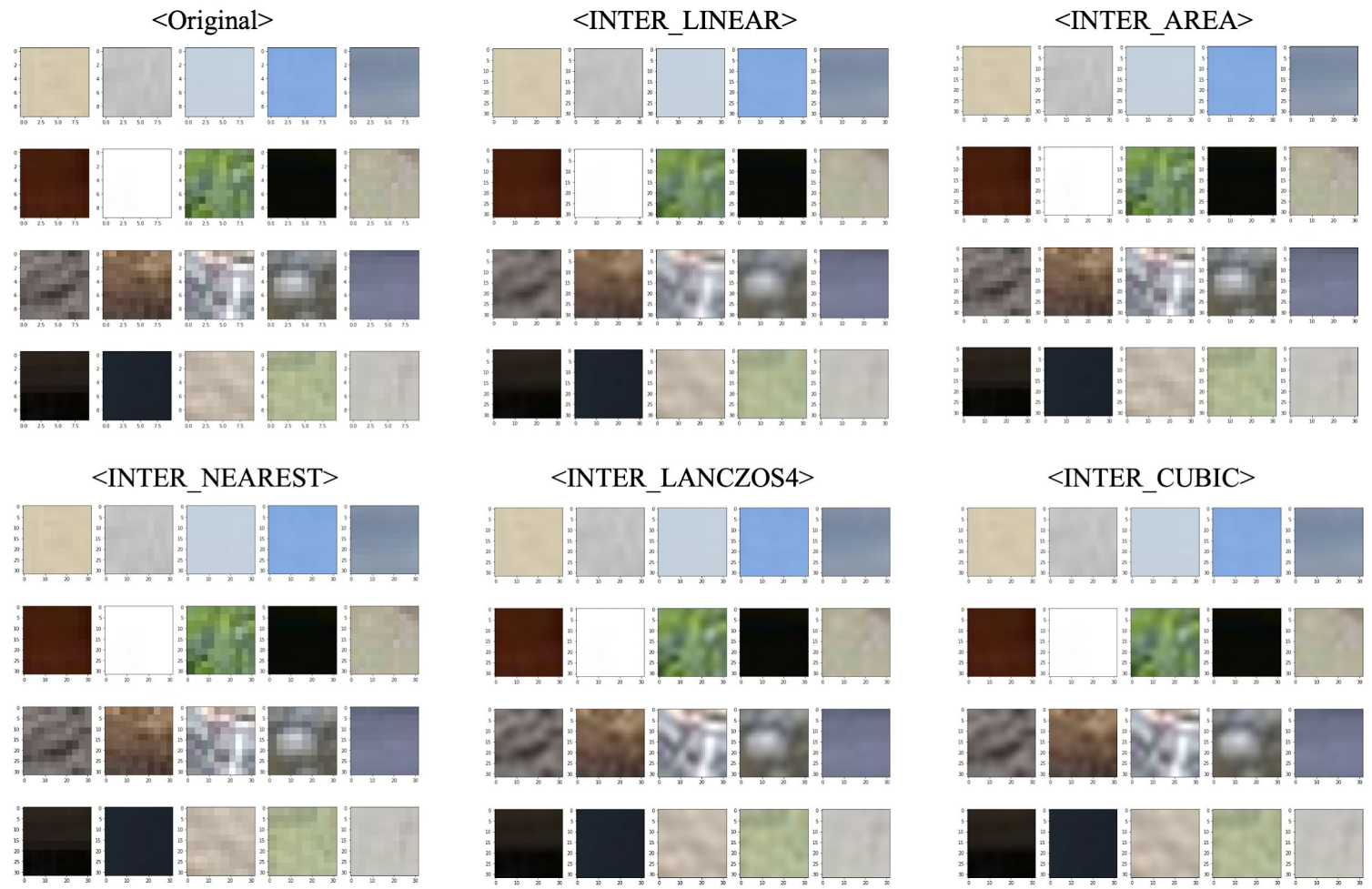
EfficientNet



Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." *International conference on machine learning*. PMLR, 2019.

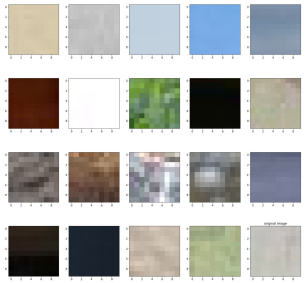
Scale the image

10 X 10 □ 32 X 32

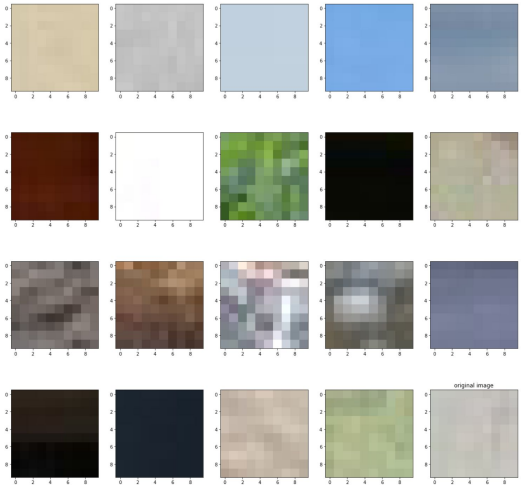


INTER_NEAREST – a nearest neighbor interpolation
INTER_LINEAR – a bilinear interpolation
INTER_AREA – resampling using pixel area relation
INTER_CUBIC – a bicubic interpolation over 4 X 4 neighborhood
INTER_LANCZOS4 – a Lanczos interpolation over 8 X 8 pixel neighborhood

Summary

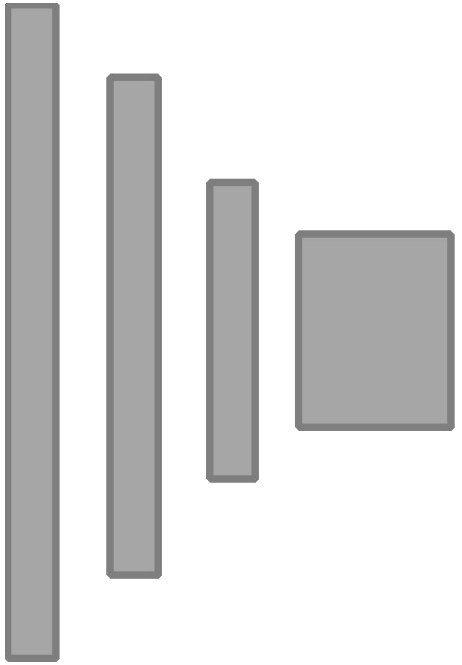


10 X 10 image

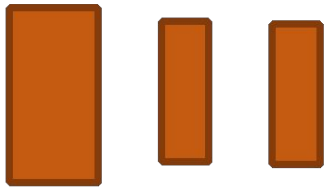


32 X 32 image

Resize using Interpolation



Pretrained model



Replaced Layer

Class 1
Class 2
Class 3
Class 4
Class 5

EfficientNet

Result

EfficientNetB1

Batch size – 32
Learning rate – 0.0001
Optimizer – Adam
Interpolation – INTER_LINEAR

	Precision	Recall	F1-score	Support
1	64.46	53.04	58.19	25900
2	69.97	75.02	72.41	33700
3	84.72	88.01	86.33	34000
4	57.42	52.22	54.69	35250
5	68.25	75.34	71.62	38800
Accuracy			**69.54**	167650
Macro avg	68.93	68.72	68.65	167650
Weighted avg	69.07	69.54	69.13	167650