Task 1 – Rohit Garg

1. Objective

To train a multi-class classifier on the dataset to predict the probability of **State**.

2. Data (Code 1 & 2)

There are 4,293 rows and 97 columns. The data is scaled so that Principal Component Analysis can be done. To scale the data standard scalar is applied (**S = (X-mean)/std_dev)**). Missing value imputation is done post standardizing the variable. The missing data is imputed with the mean.

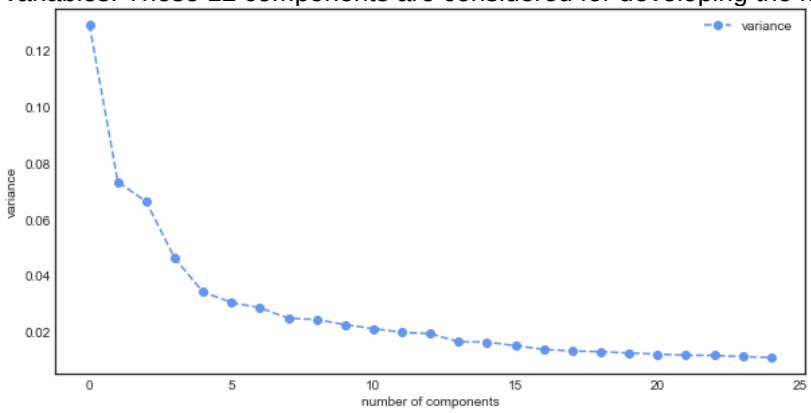
It was observed that for 3 variables there is no variance in the data. These variables are dropped from the dataset

2.1. PCA (Code 2)

Dimension reduction is done by creating new variables (called components). It is done during pre-processing of data for predictive models or forecasting. The reasons for doing dimension reduction include:

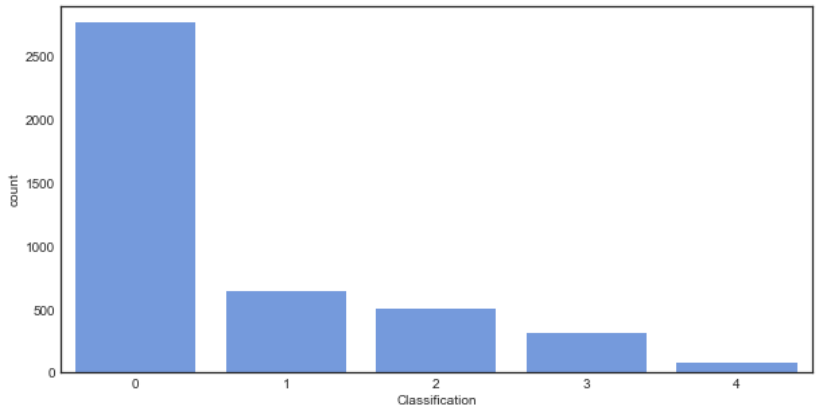
- Multi-collinearity issues
- Computational issues of large number of predictors
- Noisy models due to over fitting

Principal Component Analysis (PCA) is an algorithm for dimension reduction. It linearly transforms the variables into components. Components can be determined by the percentage of variance explained. Original data has 92 variables. PCA is used to create 12 orthogonal components from these variables. These 12 components are considered for developing the model



2.2. Dependent (Code 1)

It is observed that there are 2,757 **normal (0)**, there are 646 **network delay kanban api gateway (1)**, there are 500 **network delay kanban command service (2)**, there are 310 **pod kill kanban api gateway (3)** and there are **80 cpu burn kanban api gateway (4)**



2.3. Independent (Code 2)

It is observed that there are no missing values in any of the independent variables

	count	min	mean	max	std
C1	4,293	-6.210	00	17.342	3.444
C2	4,293	-10.943	00	22.139	2.596
C3	4,293	-7.171	00	29.636	2.468
C4	4,293	-6.938	00	8.782	2.062
C5	4,293	-6.029	00	6.890	1.774
C6	4,293	-4.786	00	14.234	1.674
C7	4,293	-7.894	00	8.585	1.621
C8	4,293	-5.621	00	17.099	1.513
C9	4,293	-14.084	00	16.431	1.496
C10	4,293	-5.220	00	13.898	1.436
C11	4,293	-7.287	00	11.205	1.396
C12	4,293	-12.823	00	16.951	1.350

3. Model (Code 3)

Gradient boosting combines a set of weak learners and delivers improved prediction accuracy. The outcomes predicted correctly are given lower weight compared to the miss-classified outcomes. The hyper-parameters of this ensemble model can be divided into 3 categories:

- **Tree-Specific Parameters:** These affect each individual tree in the model.
- **Boosting Parameters:** These affect the boosting operation in the model.
- **Miscellaneous Parameters:** Other parameters for overall functioning.

The optimal hyper-parameters are determined using iterative process. The best hyper-parameters are:

- **Max depth:** 25 (maximum depth of the individual regression estimators)
- **Subsample:** 0.5 (subsample ratio of the training instances)
- **Number of estimators:** 100 (number of boosting stages to perform)

4. Performance (Code 4)

Confusion Matrix

		Predicted				
		normal	**network delay kanban api gateway**	**network delay kanban command service**	**pod kill kanban api gateway**	**cpu burn kanban api gateway**
Actual	**NORMAL**	2,757	0	0	0	0
	network delay kanban api gateway	0	646	0	0	0
	network delay kanban command service	0	0	500	0	0
	pod kill kanban api gateway	0	0	0	310	0
	cpu burn kanban api gateway	0	0	0	0	80

- The model has an accuracy of **1.0** and F1 score of **1.0**
- The model has AUROC of **1.0** and Gini of **1.0**

5. Conclusion

Based on the performance it can be concluded that the model performs a good job at classifying the State.