Time Series Anomaly detection system

Aims -: The main aim of this problem to detection of anomaly and discovering the pattern that do not the conform that expected behavior. And also find or rare events in data streams, and referred to as an anomaly events.

Objectives -: The objective of Time series anomaly detection is to identifies the behavior of (target) binary classification variables [0 and 1] is anomaly or not.

Tools -: In this project we use numpy, pandas, seaborn, matplot, and keras.

Numpy -: Numpy is a Python library used to work with arrays that also works with linear algebra, Fourier transforms and matrices.

Pandas -: Pandas is defined as an open source library that provide high performance data manipulation in python. The name of pandas is derived from the word panel data, which means econometrics form multidimensional data. It used for data analysis in python.

Seaborn -: Seaborn is a statistical graphics library in Python Seaborn helps you explore and understand your data.

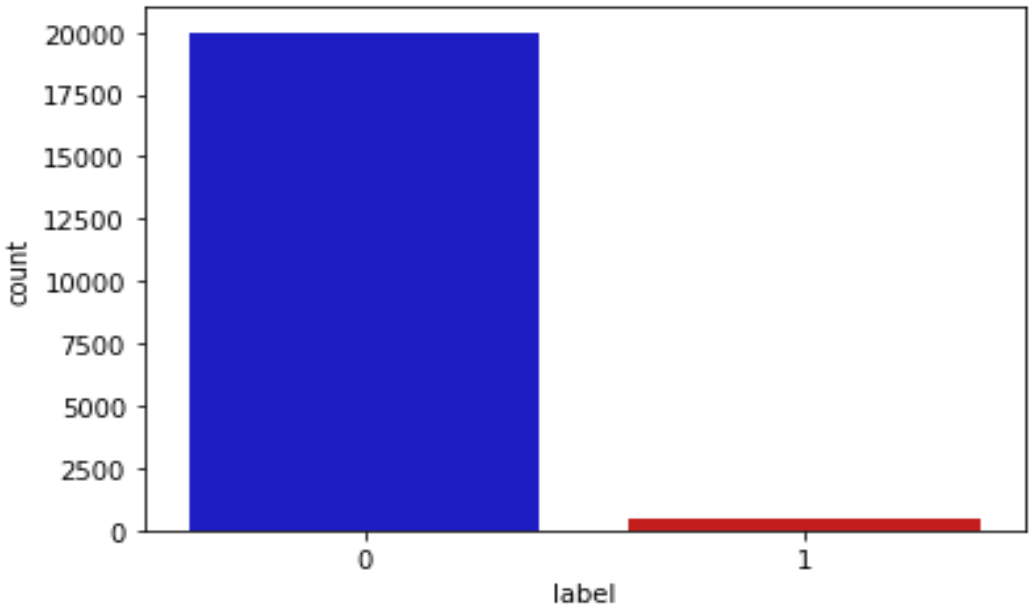
Matplotlib -: MatplotLib is a Python library defined as a multi-platform data visualization library built on a numpy array. It can be used in Python scripts, shells, web applications and other graphical user interface toolkits.

Keras -: Keras is a high-level API of Tensor Flow that provides an easy, highly productive interface for solving machine learning problems with a focus on modern deep learning..

Data Set -: The shape of the data set is 509632 rows and column 12. I did not take the 509632 rows and 12 columns because the number of majority class was too low from minority class because our model was getting over fitted and fluctuating on that over all data set. Later I selected randomly 20000 number of rows and 12 columns (v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11 and label where label column represent the Normal Class and Anomaly Class).

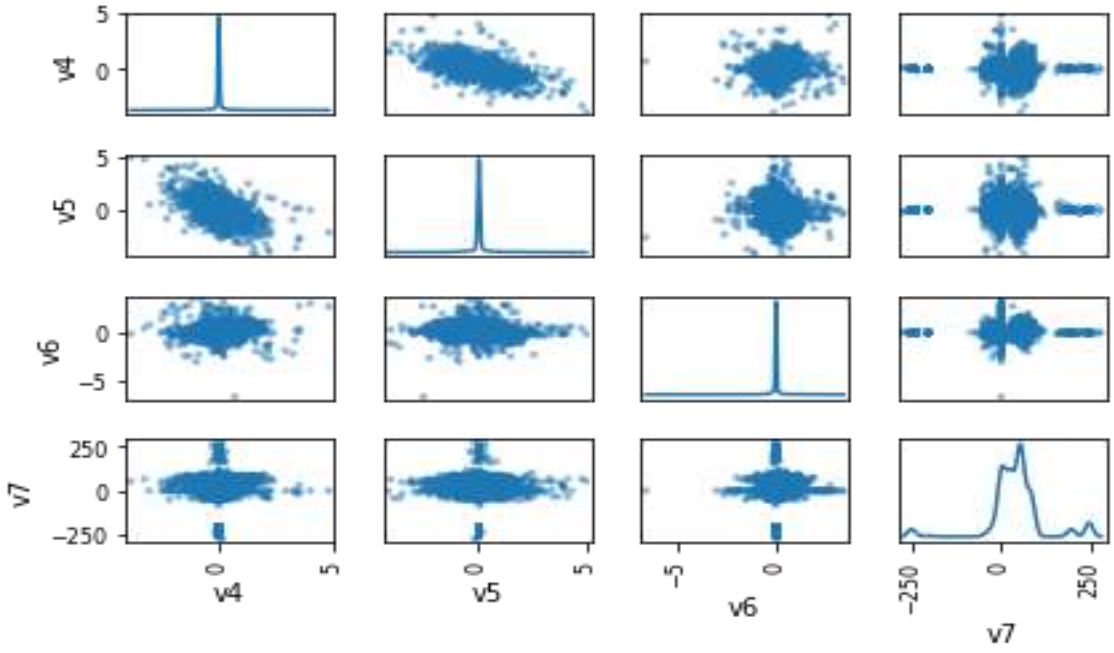
Data Analysis -: In the data analysis I implemented count plot, scatter plot, lmplot, and heat map.

Count Plot -: The below count plot diagram show the number of Normal Class and Anomaly Class where 0 is Normal Class and 1 is Anomaly Class.

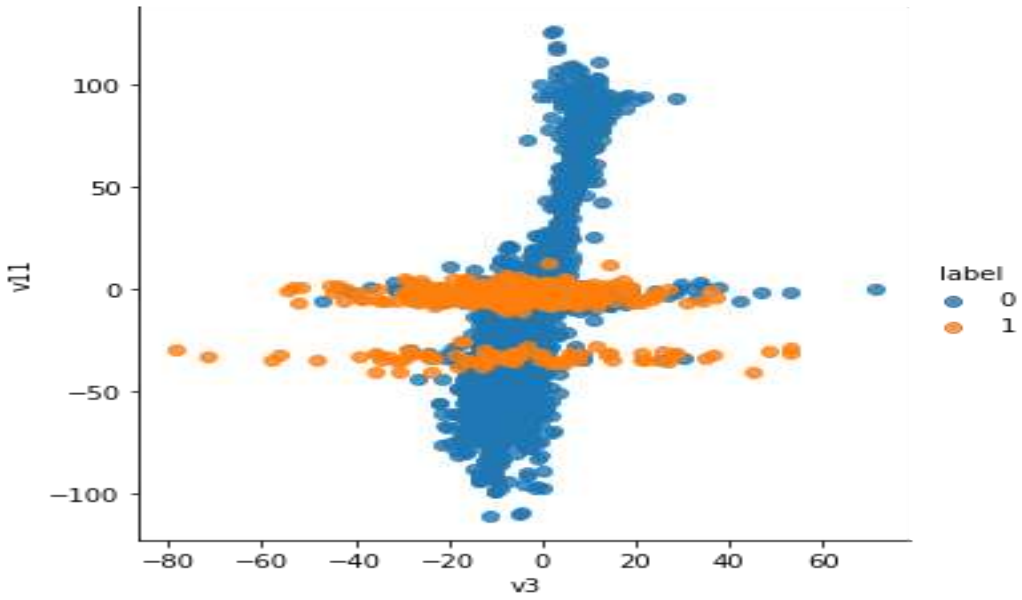


Above plot represent the 0 is normal class and 1 is Anomaly class

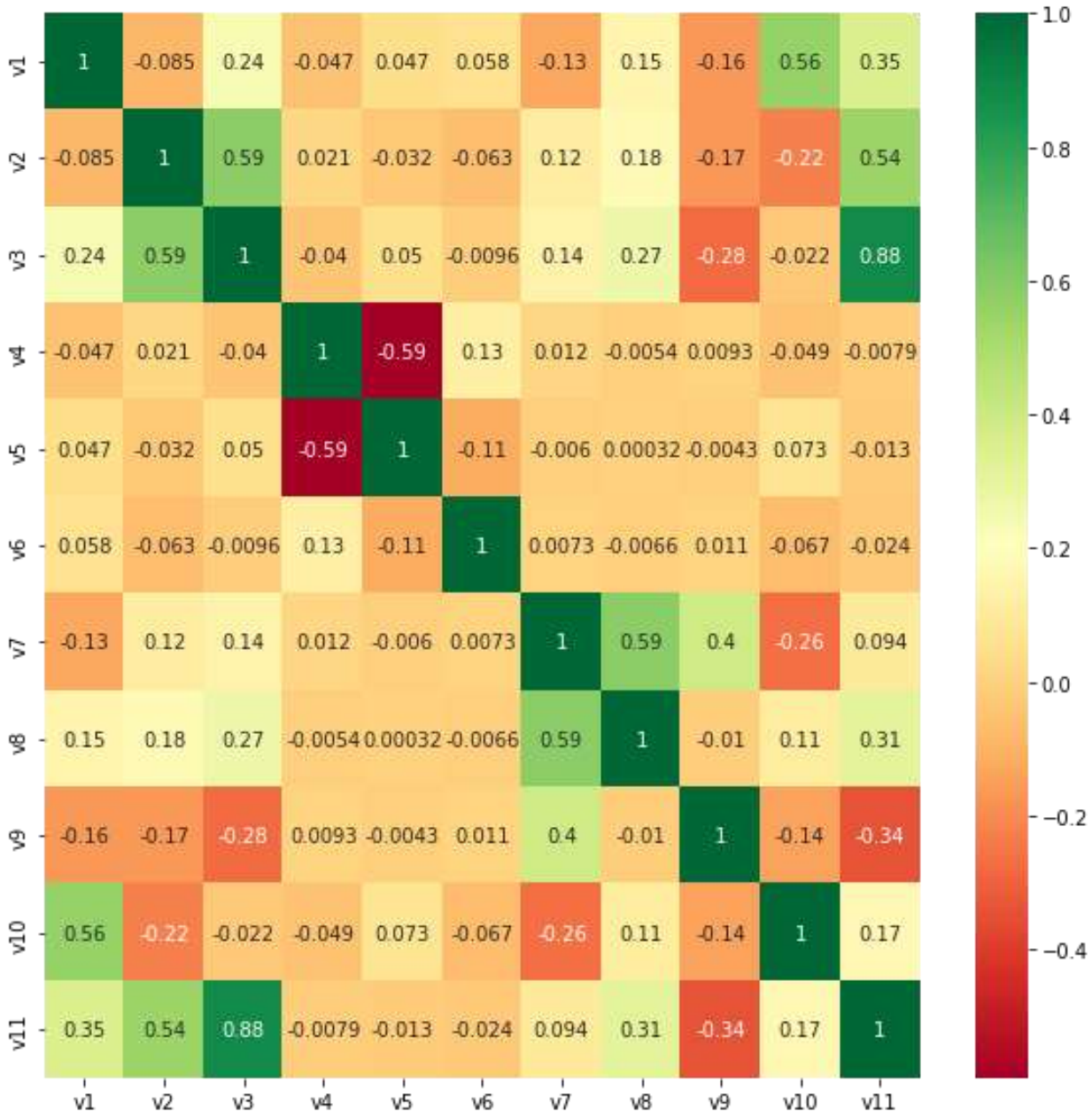
Scatter Plot -: In the below diagram, in the following matrix scatterplot, diagonal cells show histograms of the first four variables (v4, v5, v6, v7). In this matrix scatterplot, diagonal cells show histograms of each of the variables. Each off-diagonal cell has a scatterplot of v4 to v7. For example, the second cell in the first row is the scatterplot v4 (y-axis) versus v7 (x-axis).



Lmplot -: In the below diagram, the following scatterplot of v3 vs v11 shows that cultivar 1 has a lower value for v3 than cultivar 0.



Heatmap -: The below heatmap plot show the v2 and v7 are strongly correlated, and other variables are not strong correlated each other.



Min Max Scaler -: I have applied the Min Max scalar to normalize time series inconsistency detection data sets and direct input variables. I have used the default configuration and scale values for 0 and 1 range. A MinMaxScaler instance is defined with the default hyper parameters. I call the fit () function and pass it into our individual variables to create a converted version of our dataset.

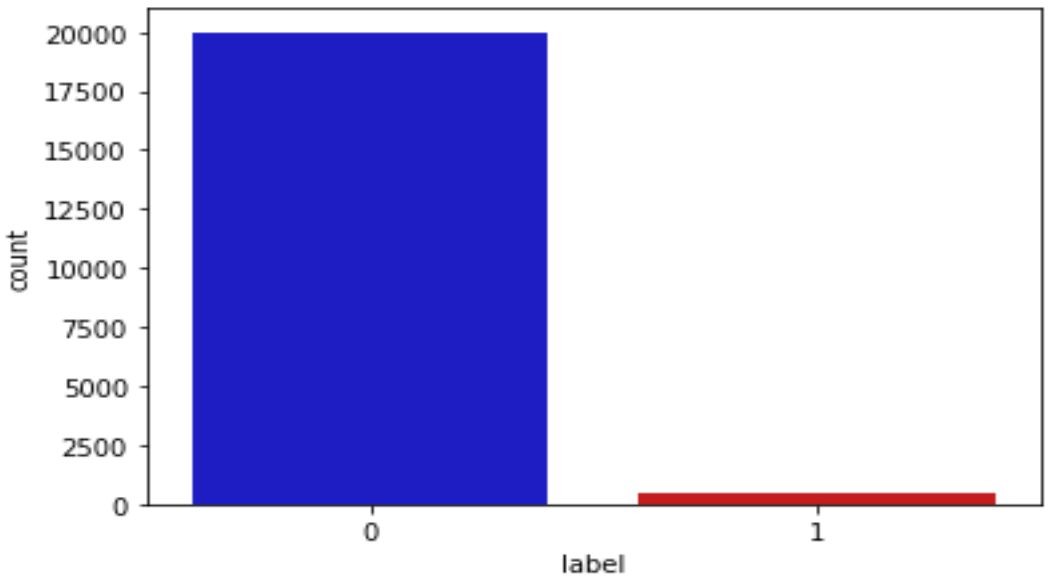
Random Oversampling -: I have use Random Oversampling method. Random Oversampling involves randomly selecting example from the minority class with replacement and adding them to the training data set. Random under sampling involves randomly selecting examples from the majority class and deleting them from the training data set.

Random Oversampling with Imblearn

The fight against imbalanced data is about creating new patterns among minority groups. The most naive strategy is to randomly generate new samples by replacing the currently available samples. Random over sample offer.

When I observed in one class is higher than the observation in other class, than there exist a class imbalance. To detect time series anomaly detection. As you can see in the below graph anomaly class around 443 when compared to the normal class around 20000.

Imbalance Count Plot graph



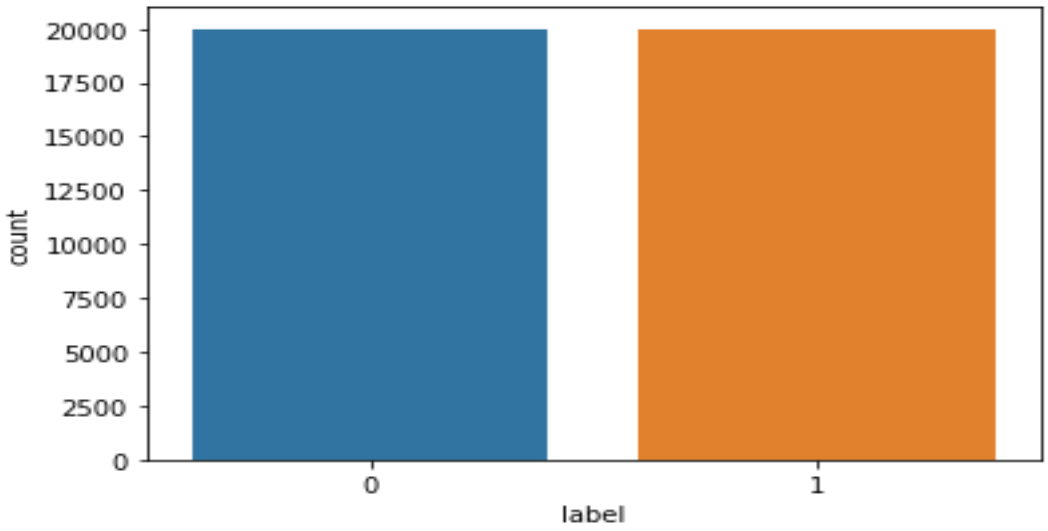
In the above figure represent the 0 is Normal Class and 1 is Anomaly Class

It has been observed that our target (label) class has an imbalance.

Later I try to implement the Up sample the data so that the minority class matches with the majority class.

After up sampling, the distribution of class is balanced, see in below graph

Balance Count Plot graph



0: Normal Class and 1: Anomaly Class

Deep Neural Network -: A neural network consists of many connected units called nodes. It is the smallest part of the neural network and acts as neurons in the human brain. When a neuron receives a signal, it initiates a process. Based on the input receiver, the signal travels from one neuron to another, creating a complex network that learns from feedback.

The Nodes are grouped into layers. The task is solved by processing at different levels between the input and output levels. The higher the number of layers to be processed, the deeper the network will be, hence the term deep learning.

Keras -: Keras runs on top of open source machine learning libraries like TensorFlow, Theano or Cognitive Toolkit. TensorFlow is one of the most famous symbolic mathematics libraries used to build neural networks and deep learning models. TensorFlow is very flexible and the primary benefit distribution is computing.

Keras is based on a minimal structure that provides a clean and easy way to build deep learning models based on TensorFlow or Theno. Keras is designed to quickly define deep learning models. Keras is a great choice for deep learning applications.

Analysis of the existing Deep Neural Network (DNN) -: I have build a deep neural network to predict the time series anomaly detection with keras.

Performance Matrix -: I use the performance matrix to evaluate the model and calculated the accuracy, precession, recall, F1-Score.

***Accuracy* –** Accuracy is a metric for evaluating the number of correct prediction divide the total number of correct prediction. The formula of accuracy is-

Accuracy = Number of correct Prediction / Total number of correct predictions.

$$\text{Accuracy} = TP + TN / TP + TN + FP + FN$$

Where TP = True Positive, TN = True Negative, FP=False Positive, FN= False Negative.

Precession - Precession is to calculate the ratio between the numbers of positive samples correctly classified to the total number of samples classified as a positive. The formula of precession-

$$\text{Precession} = \text{True Positive} / \text{True Positive} + \text{False Positive}$$

$$\text{Precession} = TP / TP + FP$$

Where TP = True Positive and FP = False Positive.

Recall - Recall is calculated by the number of positive samples divide by the total number of positive samples and negative samples. The recall measure the model detect the positive samples. The formula of Recall-

$$\text{Recall} = \text{True Positive} / \text{True Positive} + \text{False Negative}$$

$$\text{Recall} = TP / TP + FN$$

Where TP = True Positive and FN=False Negative

F1-Score - F1 score combine the precession and recall metrics into single metric. F1 score has been designed to work well on imbalanced data. The formula of F1-score is -

$$F1\ score = 2 * \frac{Precision\ * Recall}{Precision + Recall}$$

I implemented the four best model like best_model0, best_model1, best_model2, best_model4.

Best_model0 -: batch_size 16 and number of epochs in best_model0 50. Our model uses 32 density layers and the model attached to the relu activation function (Rectified Linear Unit is the most commonly used activation function in deep learning models. For positive value x it returns that value, so it can be written as f (x) = max (0, x)). And the optimizer 'sgd'.

The performance of best_model0 is listed in below table1.

Model Name	Accuracy (%)	Precession (%)	Recall (%)	F1-Score (%)
best_model0	0.99	0.99	0.99	0.99

Table1

best_model1 -: best_model1 has batch_size 16 and number of epochs 50. And our model uses 12 dense layers and model connected with relu activation function (Rectified Linear Unit is the most used activation function in deep learning models. The function returns 0 if it receives a negative

input , but for any positive value x it returns that value. So it can be written as f(x)=max(0,x)). and the optimizer 'Adam'.

The performance of best_model1 is listed in below table2.

Model Name	Accuracy (%)	Precession (%)	Recall (%)	F1-Score (%)
best_model1	0.98	0.97	0.98	0.98

Table 2

best_model2-: best_model2 has batch_size 16 and epoch number 50. and our model uses a combined model with 128 density layer and relu activation function (rectified linear unit is the most commonly used activation function in deep learning models. The function returns 0 if it receives negative input. But for any positive value x it returns that value, so it can be written as f (x) = max (0, x), and use the optimizer 'adam'.

The performance of best_model1 is listed in below table3.

Model Name	Accuracy (%)	Precession (%)	Recall (%)	F1-Score (%)
Best_model2	0.99	1.0	0.99	0.99

Table 3

best_model4-: best_model4 has batch_size 16 and epoch number 50. And our model uses a combined model with 16 density layer and relu activation function (rectified linear unit is the most commonly used activation function in deep learning models. The function returns 0 if it receives negative input. But for any positive value x it returns that value, so it can be written as f (x) = max (0, x), and use the optimizer 'sgd'.

The performance of best_model1 is listed in below table3.

Model Name	Accuracy (%)	Precession (%)	Recall (%)	F1-Score (%)
Best_model2	0.96	0.92	1.0	0.96

Table 4

Analysis of Designed Deep Neural Network (DNN) -:

In the below table4 in comparison of three model our two model give better performance from another model.

Comparison of three model

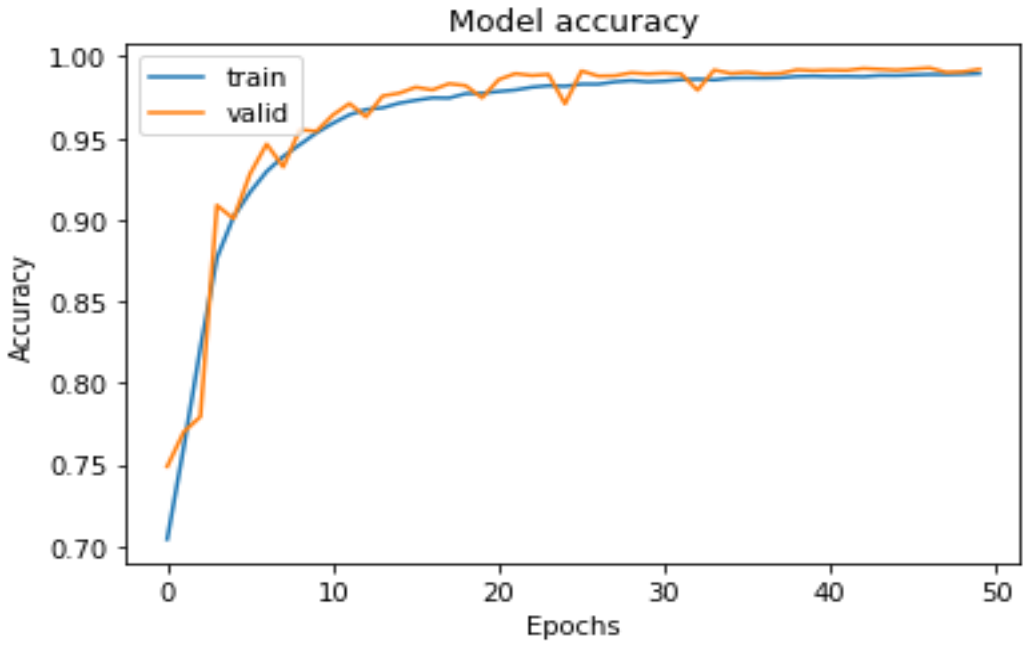
Model Name	Accuracy (%)	Precession (%)	Recall (%)	F1-Score (%)
best_model0	**0.99**	**0.99**	**0.99**	**0.99**
best_model1	0.98	0.97	0.98	0.98
best_model2	**0.99**	**1.0**	**0.99**	**0.99**
Best_model4	0.96	0.92	1.0	0.96

Table4

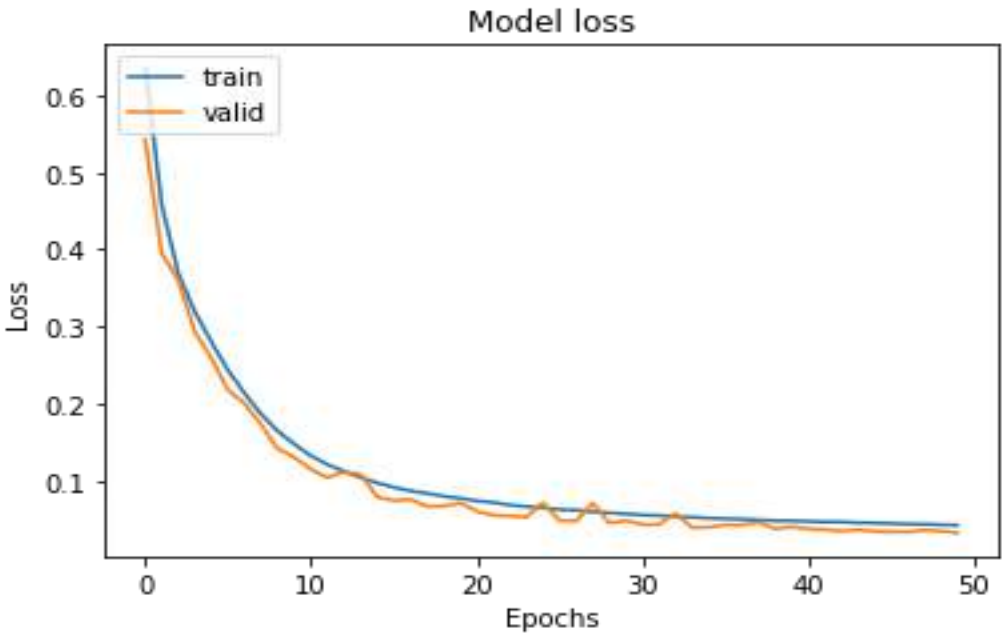
Analysis of Training Process -:

Learning Curve -: In machine learning the optimal value of the loss function of the learning curve or training curve model is evaluated against this loss function on a validation data set with the same parameters as the optimal function for the training set. The graph below shows some model learning curves for each model.

The below **best_model0** Summary is represented the model accuracy graph and model loss graph using 50 epochs.

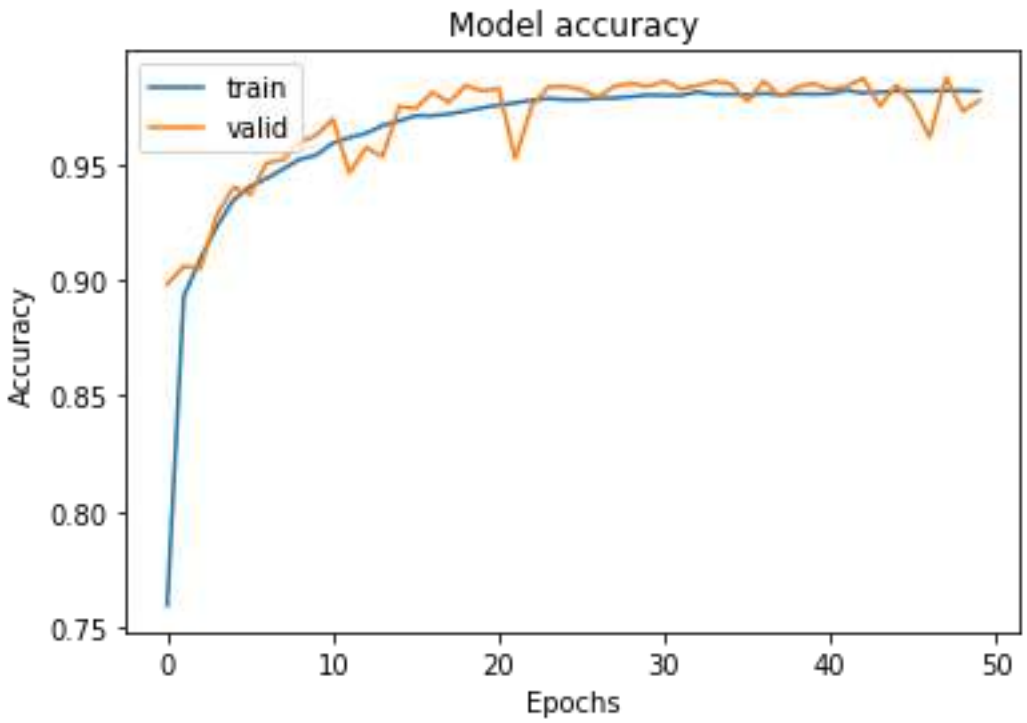


The accuracy curve both for training data as well as validation data are gradually increase from 0 to 30 epochs and 30 to 50 epochs are parallel to saturated point. We can say the model is good fit.

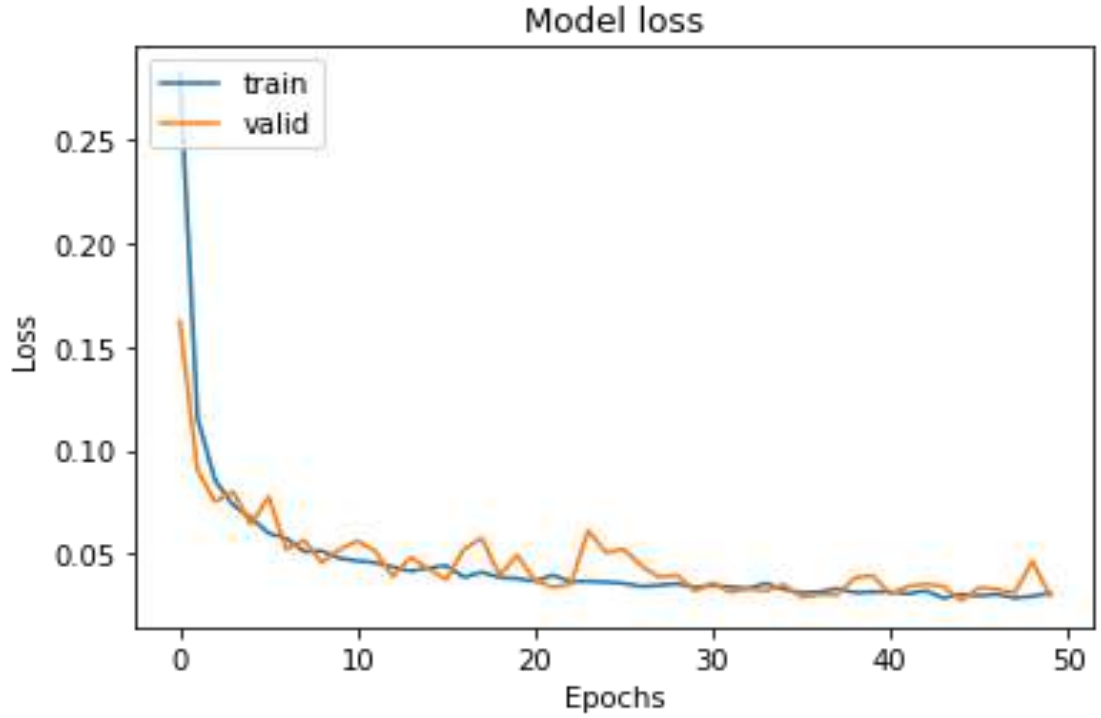


The model loss curve both for training data as well as validation data are gradually decrease from 0 to 38 epochs and 40 to 50 epochs are parallel to saturated point. We can say the model is good fit.

The below **best_model1** Summary is represented the model accuracy graph and model loss graph using 50 epochs.

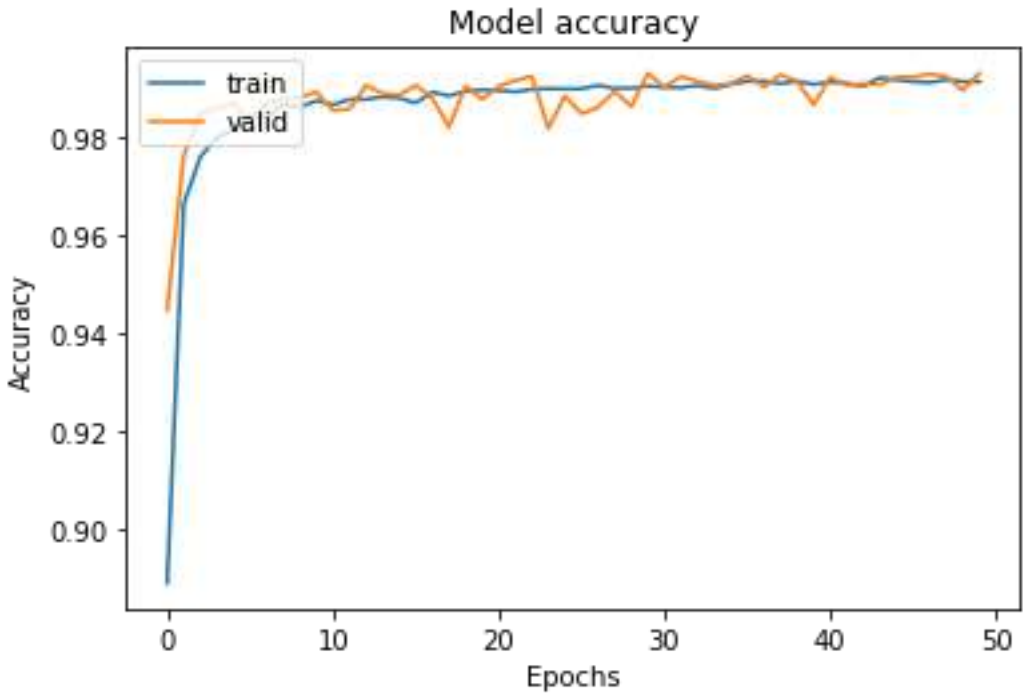


The model accuracy of training data from 0 to 50 epochs that look like a good fit for saturated point. And the validation data show the noise moment from 0 to 50 epochs around the training data.

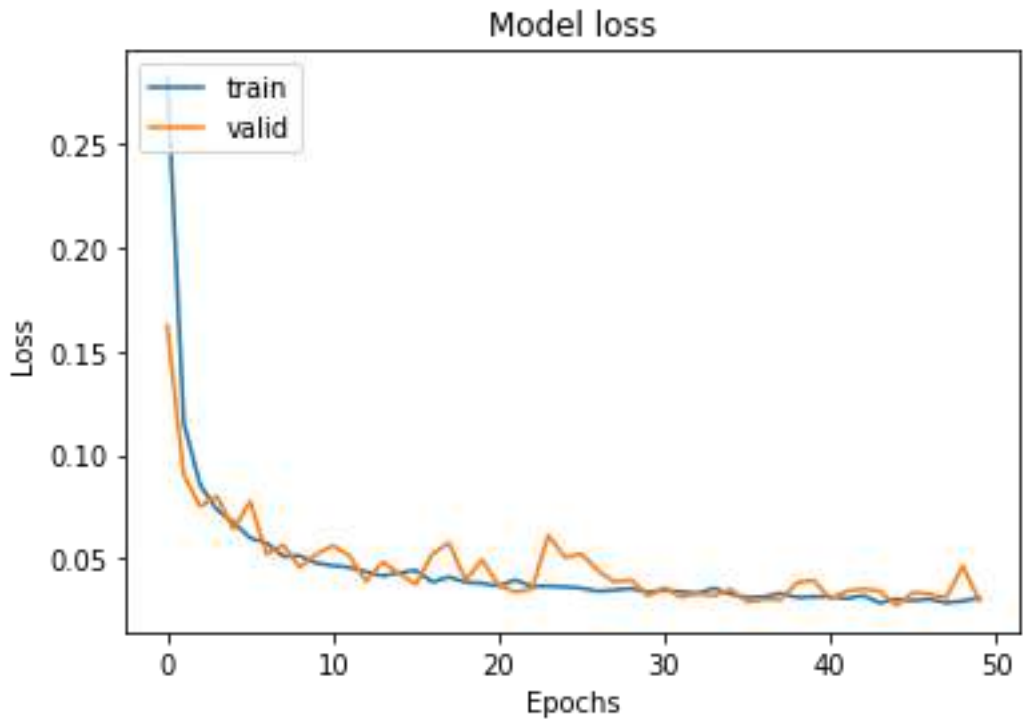


The loss of training data from 0 to 10 epochs are decreases and 10 to 50 epochs are training data is good fit and validation data from 0 to 10 epochs decreases and 10 to 50 epochs show the noise moment around the training data.

The below **best_model2** Summary is represented the model accuracy graph and model loss graph using 50 epochs.

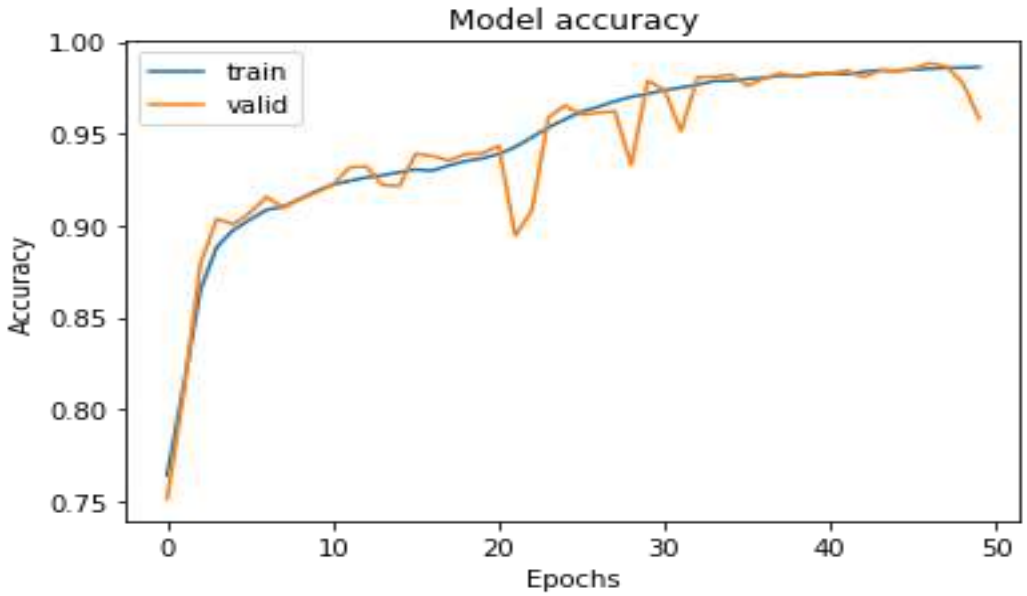


The model accuracy of training data from 0 to 50 epochs that look like a good fit for saturated point. And the validation data show the noise moment from 0 to 50 epochs around the training data.

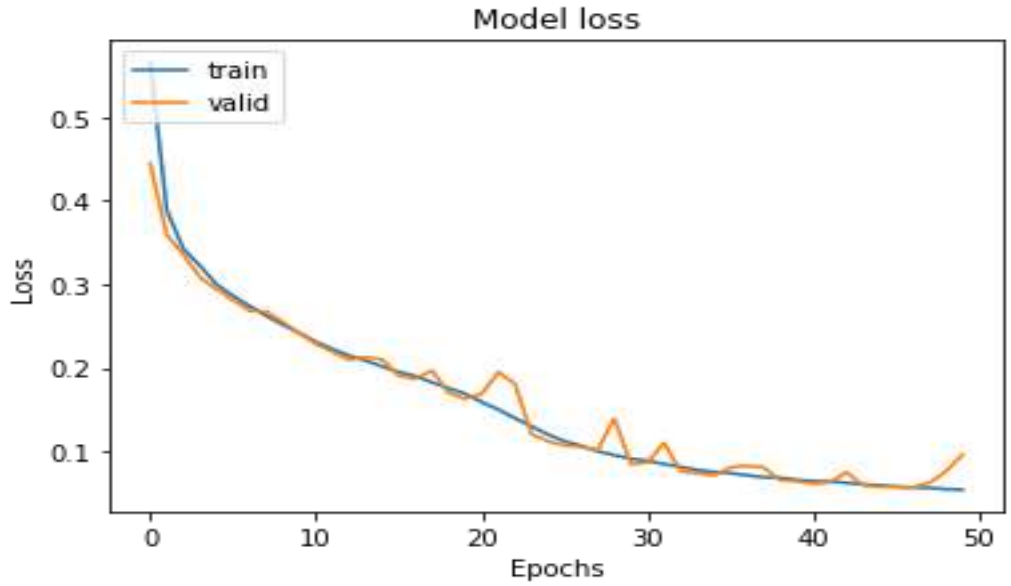


The loss of training data from 0 to 10 epochs are decreases and 10 to 50 epochs are training data is good fit and in the validation data also decrease from 0 to 10 epochs and 10 to 50 epochs validation data show the noise moment around the training data.

The below **best_model4** Summary is represented the model accuracy graph and model loss graph using 50 epochs.



The model accuracy of training data from 0 to 50 epochs that look like a good fit for saturated point. And the validation data gradually increase or decrease 0 to 30 epochs and 30 to 48 epochs are parallel from training data and validation data. And 48 to 50 epochs are validation data gradually increase.



The loss of training data from 0 to 16 epochs are decreases and 0 to 50 epochs are training data is good fit and in the validation data 16 to 50 epochs validation data show the noise moment around the training data.