place. Even though model 4 had slightly higher BIC than models 2 and 6, the model produced more sensible result.

Software release L17A Model 1 had the lowest BIC and appeared to have a good explanation when examining its plot. Thus, this model was chosen for this dataset.

5.2. State inference

Another important task after deciding which model will be used is to make an inference on the derived states because the model output does not provide any definition of these states. Therefore, an interpretation and inference of the state need to be specified. These state inferences for each dataset are shown below.

Software release L16A (see Figure 4.7)

- State1

  Despite periods where there are a decrease in CPU utilization value, two peaks which are higher than 300 clearly imply that test cases in this state perform badly. Thus, the state is defined as a *Bad* state.

- State2

  The state appears to be a *Good* state as there are many periods where test cases have low value in the CPU utilization.

- State3

  The performance for test cases in this state can be viewed as a *Good* state as well. However, State2 seems to capture the period where test cases have lowest values better than State3, and State3 has two periods of increasing to higher CPU utilization values. The state is, therefore, said to be a *Normal* state.

Software release L16B (see Figure 4.8)

- State1

  There are many high peaks occur in this state. Moreover, test cases tend to increase the value of the CPU utilization when they remain in the state. A *Bad* state is adequate for defining behavior in this state.

- State2

  The state could be characterized as a *Good* state. A decreasing pattern of the CPU utilization value can be seen in most of the period.

- State3

  The CPU utilization value for test cases in this state are either high or low. The state contains the period of positive and negative peaks and does not appear to have a specific characteristic for the state of the CPU. For this reason, the state is said to be a *Normal* state.

Software release L17A (see Figure 4.9)

- State1

  There a *Normal* state

- State2

  The period for this state is short and a decreasing pattern a *Good* state

- State3

  The only peak in the time series data is in this state. Because of this, a state is labeled as a *Bad* state.

5.3. General discussion

A thorough search of relevant literature yielded that this thesis work might be the first time that Markov switching model has been applied to the specific type of data – CPU state. In previous work, this model was mainly implemented in finance or signal processing. Researches and works on that matter were used as an inspiration; but because of some differences, the procedure is slightly changed. A large amount of time was spent on understanding the data, examining which variables had a significant impact on the CPU utilization, and determining which methods would provide the best possible outcome for this problem. Besides, after deciding on the most promising method, a lot of effort was invested studying implemented algorithms in the R package as well as modifying code as necessary.

In this study, *NodeName*, which is used to execute test cases, was assumed to be indifferent i.e., performance of test cases are the same regardless of the machine. Therefore, selecting a minimum value of the CPU utilization of the test case for each software package in data preprocessing step is acceptable.

Above all, one has to understand and accept a risk when deciding on the number of states as there is no guarantee how many states would yield the best outcome. This risk is also applied to a situation when determining the number of switching coefficients in the model. In sec. 4.1, the number of chosen states after applying Markov switching model to each dataset was three. The results from the model with two states offered less details and did not seem to cover all happening situations. Furthermore, all graphs of the two-state model had one state with a considerably long period and another state with a fairly short period. There is a problematic interpretation when trying to make a rough inference on the derived states. Despite higher BICs for the dataset of the software release L16B and L17A, three-state model provides more interpretable plots and the overall better fit to the time series data. This three states also meets with a prior defined state in the thesis objective. Note that four-state Markov switching model for each software release is also tested in the analysis. Apparently, defining four states caused more difficulty when fitting the model and making an inference on the states. The results were rather poor compare to the model with two and three states.

In sec. 4.2, a hypothesis was made – test environment variable was possible to have a switching effect. The reason for that is because local events in *EventPerSec* is essential for each test case and should have flexible values in each of the fixed number of state. Having a constant value for these variables would be inadequate to characterize the whole time series data. On the contrary, the test environment does not necessarily require a switching mechanism. Although these test environment variables affect the CPU utilization, they do not need to have different values in each state.

Another thing worth mentioning is that when modeling the Markov switching model, it is better to try estimating a simple model first. The model's size grows exponentially according to the number of states k, and the number of predictor variables n. In addition, if all coefficients have switching effects, the model will have more parameters to be estimated. The function in the package will try to fit the model and estimate coefficients, but there is no guarantee that the obtained results of the likelihood will be a global maximum. As a consequence, the values of estimated coefficients cannot be completely trusted. Perlin (2015) suggested any model with $k > 3$ and $n \geq 4$ is not recommended. This is also one of the main reasons why not including all local events which were generated in each test case, and not making the model to have all switching coefficients.

As mentioned previously in sec. 4.1, one variable in the dataset of the software release L16A was excluded from the regression model. Fifty-seven observations in this dataset is used to train the Markov switching model. The dataset (or the number of test cases in the software release) is rather small which increases an occurrence of singularity. Therefore, unless there is more data, it is better to drop the variable from the regression model before doing a further analysis.

For each software release, each state in the model have a significantly high r-squared which are greater than 0.9 (see Appendix C). R-squared is an intuitive measure to determine how well the model fits the data. In general, the higher r-squared is the better. Nevertheless, high r-squared does not necessarily mean that the model fits the data well. R-square cannot determine whether the model is adequate or not. In order to assess this, the residual analysis is required.

One reason for getting a high r-squared is because of including many predictor variables to the model. R-squared increases when there are more terms in the model.

Another reason, which is somehow a consequence of adding too many predictor variables, is that the model might be overfitting the data. As a result, the model produces misleading high r-squared.

A Q-Q plot is a visual check for an assumptiosteady n of normality which provides a rough idea whether or not the assumption is plausible. From residual analysis in sec. 4.3, it revealed that residuals did not completely follow the normal distribution for all three datasets. This is not surprising as real data rarely have a well-defined structure. Since the Markov switching model assumed normal distributed residuals, the chosen models might have lesser in credibility in the results. However, recall that the dataset used in this thesis is not sufficiently large. With small samples, the Q-Q plot is often less clearer and difficult to spot a basic feature as it might represent more noise.

More importantly, the results of the Markov switching model could be affected by various types of bias. First of all, training the model with a small dataset. Due to a lack of data, the model will be less effective or unable to capture the behavior for new observations that lie outside the range of training data. For this reason, a prediction for the new observations might not be accurate as it should be. This is like what happened when making a state prediction for the software release L16A (see sec. 4.6). Second, other factors which are not considered in the model might also be the reason of causing a bias. The chosen predictor variables in this thesis are variables that have a partial prior knowledge and have been analyzed to have some significant impacts on CPU utilization. However, it is possible that there are still some explanatory information that is overlooked. Finally, selecting the number of states and switching coefficients in the model could cause a bias as well.

As described above, the small dataset is proven to cause several problems and difficulty to the analysis. The size of data is crucial in statistical analysis because more information can be extracted and used as an input for the model to learn.

A benefit from using a non-parametric analysis is that it does not require a prior assumption of a data distribution and is able to detect any type of distributional change within a time series. One noted behavior which can be seen from the results of the Markov switching model and the E-divisive method in the simulated data is that if a pattern of changing from one state to another state in the data is not obvious, the E-divisive method will be unable to detect locations of change points. For instance, the method could not identify any changes in the first hundred observations, and also at the end of the time series in the simulated Dataset 1. Besides, the duration between states also affect the ability of discovering the change point locations in the E-divisive method. To illustrate, the switching pattern is considerably difficult to notice in the simulated Dataset 2. The period of staying in the state is short and there are many switches between states occur over the period of time. The shift for the response variable is not dramatic that one can see the huge difference when there is a switch in the state. Therefore, the E-divisive method could only detect two change point locations where the shifts were obvious.

After examining the results in both simulated datasets, the E-divisive method proved to have less power in detecting changes in the data. The E-divisive method and the Markov switching model appear to discover changes at around the same time when the actual changes occur, despite some false alarms and missed detections. This suggests that for the real data where the actual state is unknown, the actual changes might occur around locations where these two methods are able to detect. In addition, there is a high probability to be an actual change of the state if these two methods can identify the change point at the exact same location.

Note that when performing a Markov switching model, variables which have influences on the CPU utilization are also included in the model. Nonetheless, the E-divisive method only considers the values of the CPU utilization. With this difference, the E-divisive method will have even less power to identify actual changes in the CPU utilization. Since there are other variables that affect the value of the CPU utilization, it is insufficient to take into account only the response variable.

The state prediction function which is an additionsteady al implemented function in the package seems to work properly. This can be observed by examining the results from sec. 4.7. The accuracy of the test set in the simulated Dataset 1 was significantly high. One reason for that might be because the simulated data had an obvious pattern in switching between states. Besides, each state remained in its own state for a while before switching to the other states. Therefore, the model can completely capture the behavior of the time series data. In contrast, the simulated Dataset 2 had several switches between states and each state did not have a long duration. Even though the pattern of the response variable is rather difficult to see in the plot, the Markov switching model still perform rather well for this dataset.