OPTICAL INTERCONNECTION DATA PERFORMANCE CLASSIFICATION USING MACHINE LEARNING

PRIYANKA KONDAPARTHI

SAN JOSE STATE UNIVERSITY

Table of Contents

I Data Description

OPTICAL INTERCONNECTION DATASET contains 2 Dimensional network data passing through multiprocessor architecture. The Raw data is not arranged properly and have an extra 5 unnamed columns with null values. The data is arranged by delimiting and eliminating the unnamed columns with null values. The dataset contains an overall of 10 attributes which consists of predictors and variables with 640 datapoints each.

A total of <u>10 attributes</u> including the input features and the output variables.

- **Predictors (5):** 'Node Number', 'Thread Number', 'T/R',
    - 'Spatial Distribution': Four types of synthetic traffic patterns, Uniform (UN), Hot Region (HR), Bit reverse (BR) and Perfect Shuffle (PS)
    - 'Temporal distribution': Message Passing protocols, these are two packet generation independent (i.e., Asynchronous) and non-independent (i.e., Client-Server) traffic sources
- **Variables (5)**: 'Processor Utilization', 'Channel Waiting Time', 'Input Waiting Time', 'Network Response Time', 'Channel Utilization'

<u>Data type</u>: the data type of input and output varies

Attribute	**Data type**
Node Number, Thread Number	Integer
T/R	Float
Spatial Distribution (UN, HR, BR, PS) Temporal Distribution (ASYN, SYN)	Object (string)
Variables	Float

<u>Missing Data</u>

By observing the information of data there is no missing Data

II Data Visualization

The spread of data is different for each of the feature as evident from histogram distributions. The feature variables 'Node Number', 'Thread Number', 'T/R', 'Spatial Distribution' and 'Temporal Distribution' data is uniformly distributed. There is no outliers for independent variables.

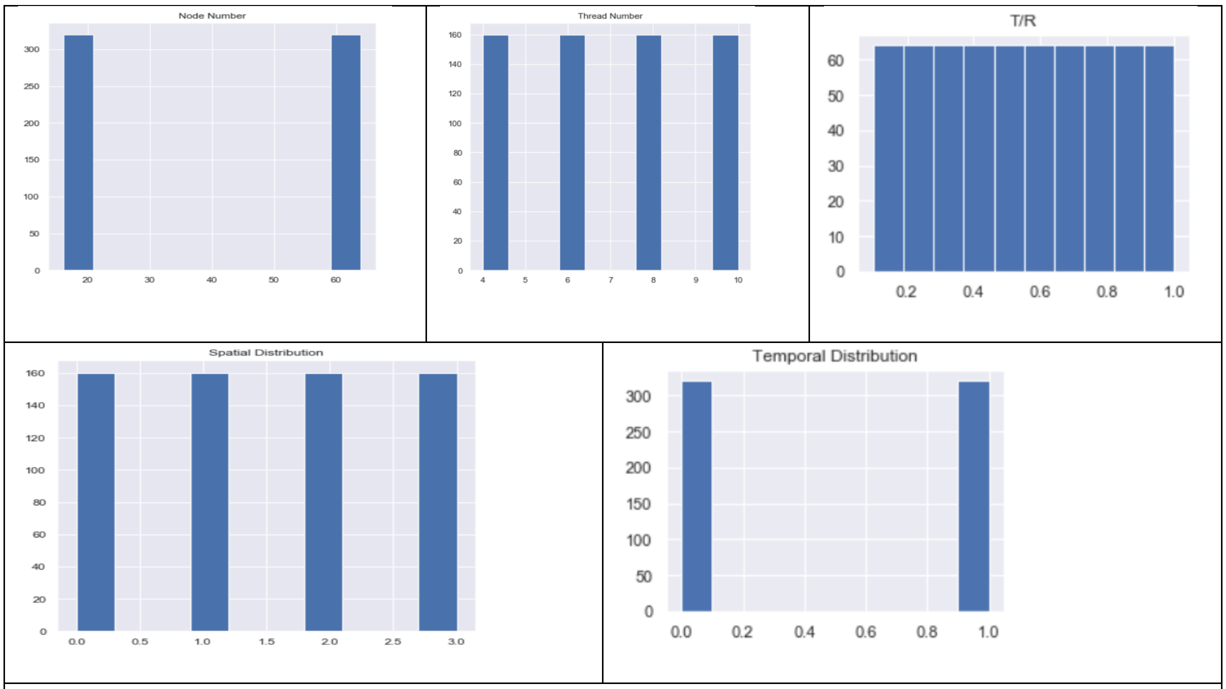


Fig 1: Histogram of the Features to understand the spread of data

The Dependent Variables distribution is described below: The output data is distributed in a random fashion Channel Waiting Time: The Distribution has 1 peak and skewed towards right. The Processor Utilization has 2 peaks in the distribution, the Network Response Time is Right skewed, and it is a unimodal. The Input waiting time is also a unimodal with tail towards right and the Channel Utilization has single peak with few data points spread to right.

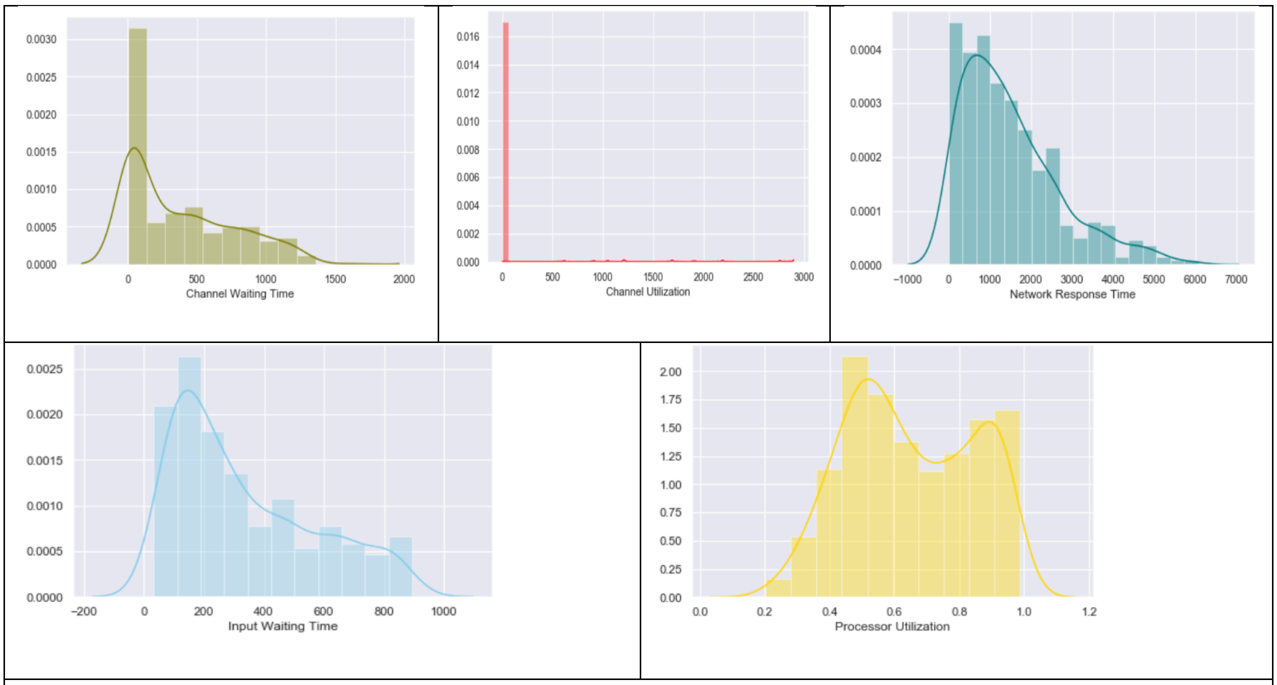


Fig 2: The Histogram plot for dependent variables

To know even more about the data spread and to check for the outliers, boxplot is taken, From the boxplot we one can say that the channel utilization and Network response time have many outliers. Also, the channel waiting time has few outliers.

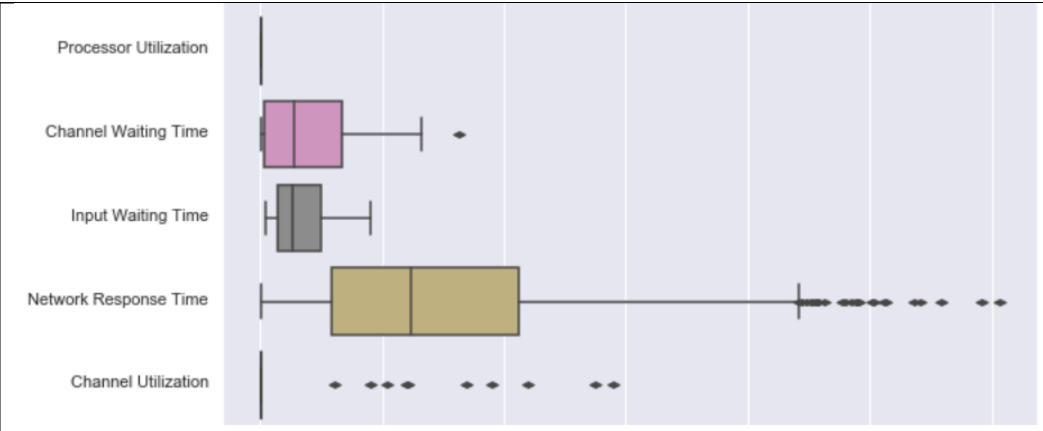
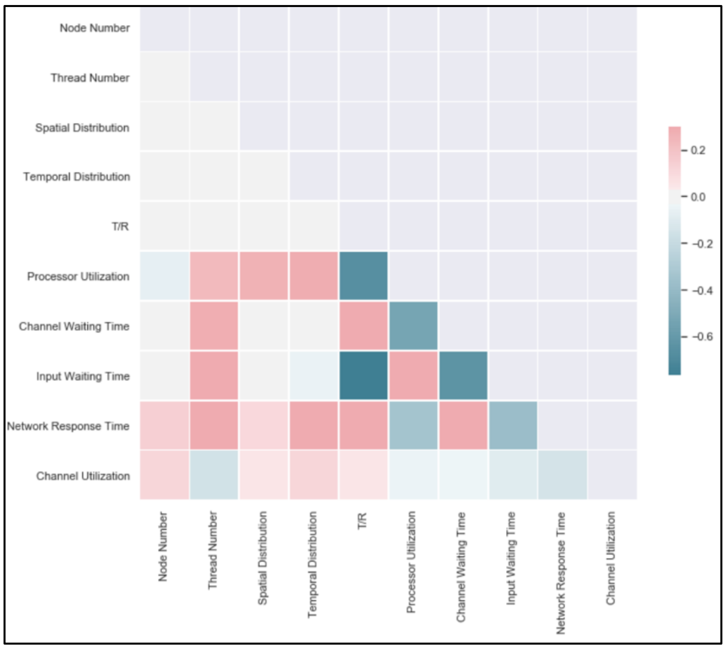


Fig 3: Boxplot of output variables to know about outliers in data.

Understanding the Correlation of the Attributes



By observing the correlation matrix the processor utilization and T/R, T/R and Input waiting are highly negatively correlated. The Thread number is slight positive correlated with the outputs Processor utilization time, channel waiting time, Input waiting time and Network Response time. Network response time is positively correlated with the inputs thread number, temporal distribution and T/R

III Data Set Cleaning

Since we do not have the any missing data, we need not eliminate the data. But we have the outliers for the dependent variables.

The total number of datapoints are 640 which are very less, so we do not remove the outliers.

IV. Related Work

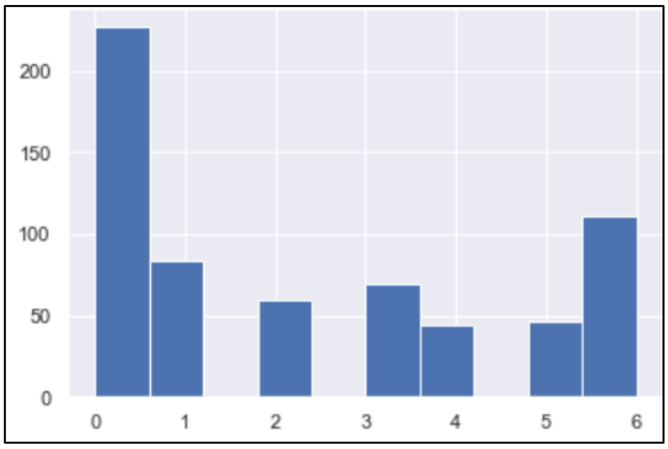
Encoding of the Input variables:

The inputs 'Spatial Distribution' and 'Temporal Distribution' are non-numeric, So we need to label encode from string to number.

Spatial Distribution (UN, HR, BR, PS) is encoded as [0,1,2,3]

Temporal Distribution (ASYN, SYN) is encoded as [0,1]

Labelling the Output Variables:



The output variables are continuous, these are to be converted into ordered categorical variables using cut points and these labelled data is the output to be classifying.

Fig 5:The Channel waiting time is labelled as (0,7) classes using cut-points

Data Pre-processing

Normalization: The inputs are spread in a wide range of 0 to 7000, Scaling is needed for the data, Inputs have been normalised using Standard scaler which maintains standard deviation of 1 and mean 0 for the feature

In the project, I used Scikit library's StandardScaler function to standardise the inputs. The input is subtracted with the mean and divided by variance, Therefore the variance of distribution is unit. Also almost all the models work well with the scaled data.

V. Feature Extraction and Feature Selection

Feature selection plays an important role in the fitting the model and to yield better accuracy and interpretability, since all predictors may not be needed for the model to be accurate. The model contains 5 predictors, and chosen one of the outputs Channel Waiting Time as the output. The inference from correlation matrix gives that the relation between the output and Thread Number, T/R has more correlation than compared to other inputs.

Observing the p-value after performing the least squares, the **p-value** with in the range 0.01 and 0.5 is for the input variables thread number and T/R ratio, but the **F-Stat** value is 56, so it means there is relation between the inputs and outputs.

Performed **Best Subset Selection** process to select the subset which has low test error, the subset with only 2 predictor (Thread Number, T/R) gave the lowest test error. Also performed stepwise selection method **forward subset selection**, observed Cp (AIC), BIC, or adjusted R2 by using each of the predictor, the performance is better with the same 2 Predictors, Thread Number and T/R. [3]

Note: Code for subset selection is taken from GitHub [3]

Also performed L1 regularization for logistic regression where the predictors Spatial Distribution and temporal distribution shrink to zero.

Ridge Regression is applied to the data which shrinks 3 features more towards 0, which are same as in lasso regression.

Therefore, the feature space is reduced from 5 Dimensions to 2 Dimensions.

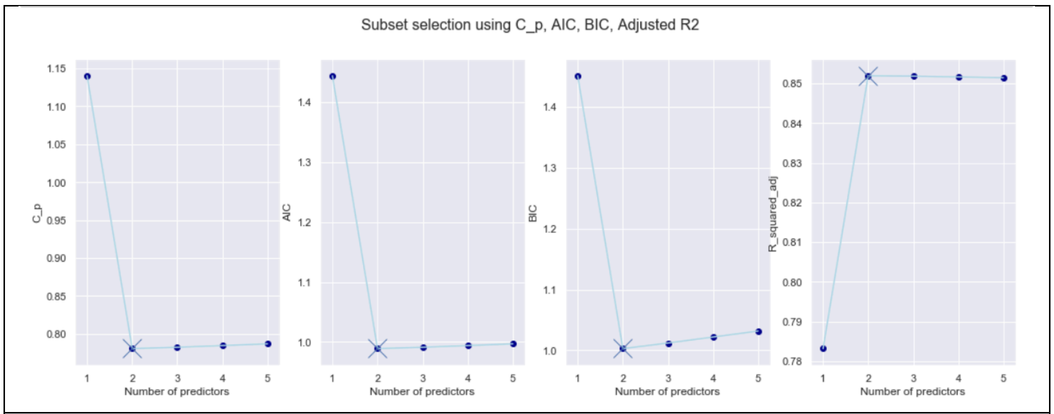


Fig 6: Cp, AIC,BIC and Adjusted R^2 for best subset.

Since Note: we have less number of features, I have developed the models using all the features and also for the Subset of features which are extracted.

VI Model Development and Fine Tuning

The example models are developed for the 5 features and the output Channel Waiting Time. The overall data is not used for training, in order to validate the models 20% of data is used as validation set and rest of the data is used for training purpose.

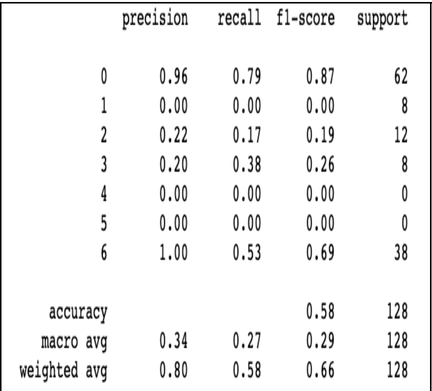
The data is shuffled and spitted using train_test_split, a cross validation function from Scikit library.

Training Data	X_train: (512,5) Y_train: (512,1)
Test Data	X_test: (128,5) Y_test: (128,1)

Table 2: Test train split

<u>Model 1: Logistic Regression</u>

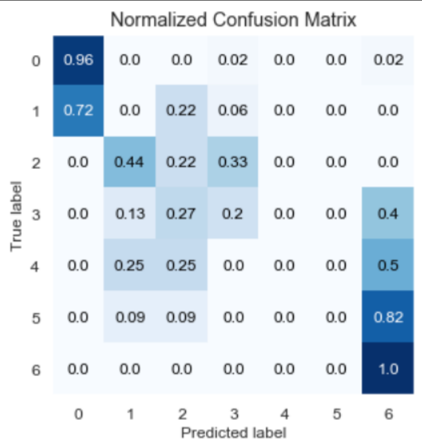
Logistic Regression classifier with linear boundary is interpretable and simple. The data is fitted to Logistic Regression classification model , The training accuracy is 60% and the test accuracy is 58%, which is less, The ROC curve also shows that only class 0 and 6 has large area under the curve, which means only class 0 and class 6 are classified well, the rest of all classes are not classified well.



```
              precision    recall  f1-score   support

           0       0.96      0.79      0.87        62
           1       0.00      0.00      0.00         8
           2       0.22      0.17      0.19        12
           3       0.20      0.38      0.26         8
           4       0.00      0.00      0.00         0
           5       0.00      0.00      0.00         0
           6       1.00      0.53      0.69        38

    accuracy                           0.58       128
   macro avg       0.34      0.27      0.29       128
weighted avg       0.80      0.58      0.66       128
```



Normalized Confusion Matrix

True label \ Predicted label	0	1	2	3	4	5	6
0	0.96	0.0	0.0	0.02	0.0	0.0	0.02
1	0.72	0.0	0.22	0.06	0.0	0.0	0.0
2	0.0	0.44	0.22	0.33	0.0	0.0	0.0
3	0.0	0.13	0.27	0.2	0.0	0.0	0.4
4	0.0	0.25	0.25	0.0	0.0	0.0	0.5
5	0.0	0.09	0.09	0.0	0.0	0.0	0.82
6	0.0	0.0	0.0	0.0	0.0	0.0	1.0

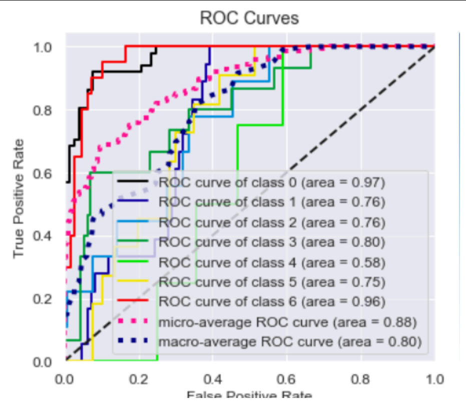
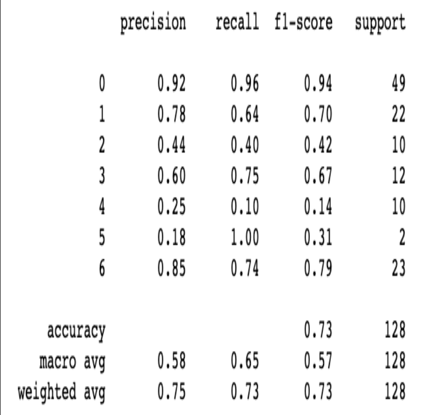


Fig 7: Performance evaluation of Logistic Regression

Model 2: LDA

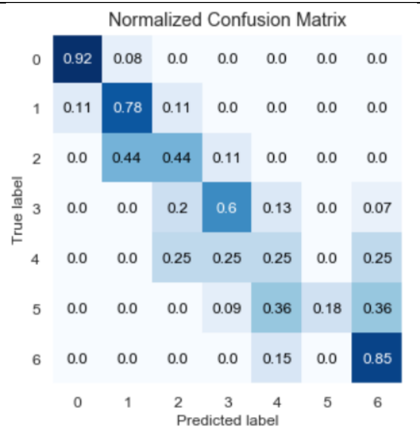
The data is fit to another linear model, **Linear Discriminant Analysis**, the accuracy is increased by 13% when compared to the Base model Logistic Regression, The training accuracy is 74% and the test is 73%, we can infer that the distribution of data might be gaussian. The class 0,1 and 6 are classified well, But the performance of the model is not good with the 70% accuracy. Therefore the model does not fit the data perfectly.



```
              precision    recall  f1-score   support

           0       0.92      0.96      0.94        49
           1       0.78      0.64      0.70        22
           2       0.44      0.40      0.42        10
           3       0.60      0.75      0.67        12
           4       0.25      0.10      0.14        10
           5       0.18      1.00      0.31         2
           6       0.85      0.74      0.79        23

    accuracy                           0.73       128
   macro avg       0.58      0.65      0.57       128
weighted avg       0.75      0.73      0.73       128
```



Normalized Confusion Matrix

True label \ Predicted label	0	1	2	3	4	5	6
0	0.92	0.08	0.0	0.0	0.0	0.0	0.0
1	0.11	0.78	0.11	0.0	0.0	0.0	0.0
2	0.0	0.44	0.44	0.11	0.0	0.0	0.0
3	0.0	0.0	0.2	0.6	0.13	0.0	0.07
4	0.0	0.0	0.25	0.25	0.25	0.0	0.25
5	0.0	0.0	0.0	0.09	0.36	0.18	0.36
6	0.0	0.0	0.0	0.0	0.15	0.0	0.85

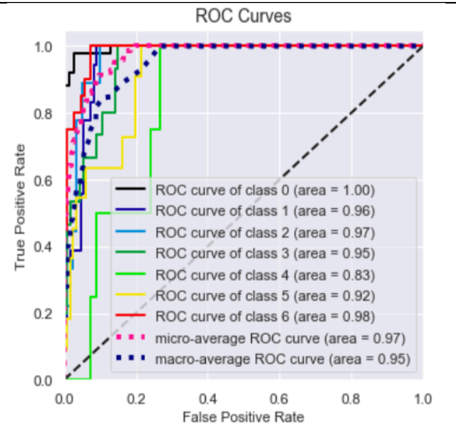
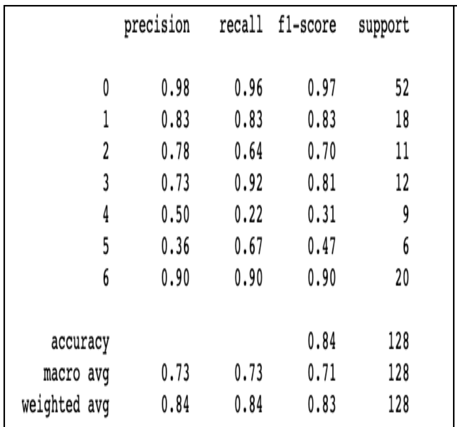


Fig 8: Performance evaluation of LDA

Model 3: QDA

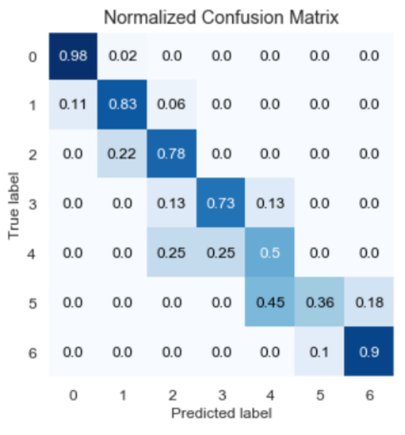
Since Logistic Regression performance is poor it is possible that there is non-linear relationship between the predictors and response. So the data is fitted to non-linear model **Quadratic Discriminant Analysis**. The test accuracy is increased to 84% which is quite a good increase in the performance. By observing the test and training performance, the data might have a normal distribution and each class has different covariance. The QDA might be performing better because there are limited number of data points (n=512).



```
              precision    recall  f1-score   support

           0       0.98      0.96      0.97        52
           1       0.83      0.83      0.83        18
           2       0.78      0.64      0.70        11
           3       0.73      0.92      0.81        12
           4       0.50      0.22      0.31         9
           5       0.36      0.67      0.47         6
           6       0.90      0.90      0.90        20

    accuracy                           0.84       128
   macro avg       0.73      0.73      0.71       128
weighted avg       0.84      0.84      0.83       128
```



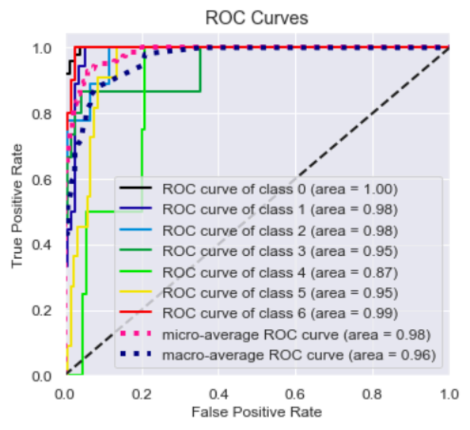


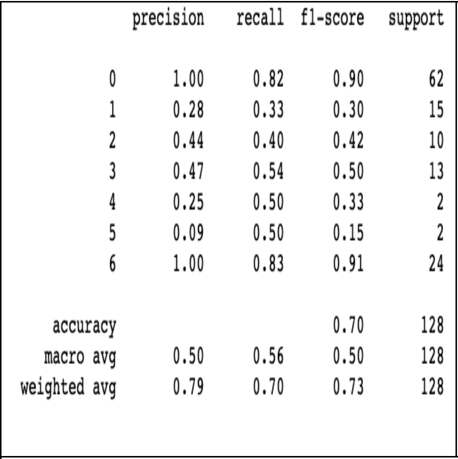
Fig 9: Performance evaluation of QDA

Model 4: KNN

As it is clear that the data is non-linear, I have performed the non-parametric machine learning model **K- Nearest Neighbour** model on the data, the tuning parameter is the K, I have tried different K, and used cross validation for the best K, the best K is 10

Hyper parameter tuning:

K = 5 over fits the data giving training accuracy 76% and test as 67%, K= 12 underfits the data making both training and test data low which are 67 and 65% each, K = 10 fits the model properly with training data as 71.5% and test data as 70%, The accuracy of 70% is not a good, we can infer that KNN is not an appropriate model for classifying the data. As KNN is a non-parametric model it needs more data to fit the model, but we have less data points. Also infer that the data might have quadratic distribution.



```
              precision    recall  f1-score   support

           0       1.00      0.82      0.90        62
           1       0.28      0.33      0.30        15
           2       0.44      0.40      0.42        10
           3       0.47      0.54      0.50        13
           4       0.25      0.50      0.33         2
           5       0.09      0.50      0.15         2
           6       1.00      0.83      0.91        24

    accuracy                           0.70       128
   macro avg       0.50      0.56      0.50       128
weighted avg       0.79      0.70      0.73       128
```

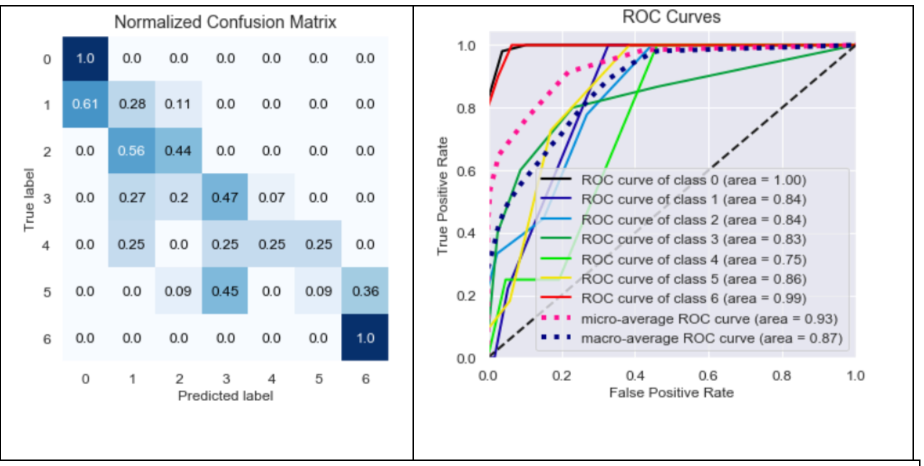


Fig 10: Performance evaluation of KNN

Model 5: DECISION TREE

One of the most simple and interpretable models, **Decision trees model** is applied onto the features, the data fits the model giving the high accuracy of 91%, We can infer and confirm that the data distribution is purely non-linear. In this model the predictor space is divided into non-overlapping J regions. The observations are predicted using most commonly occurring classes from the training set. The quality of split in our Decision trees is measured by Gini Index function.

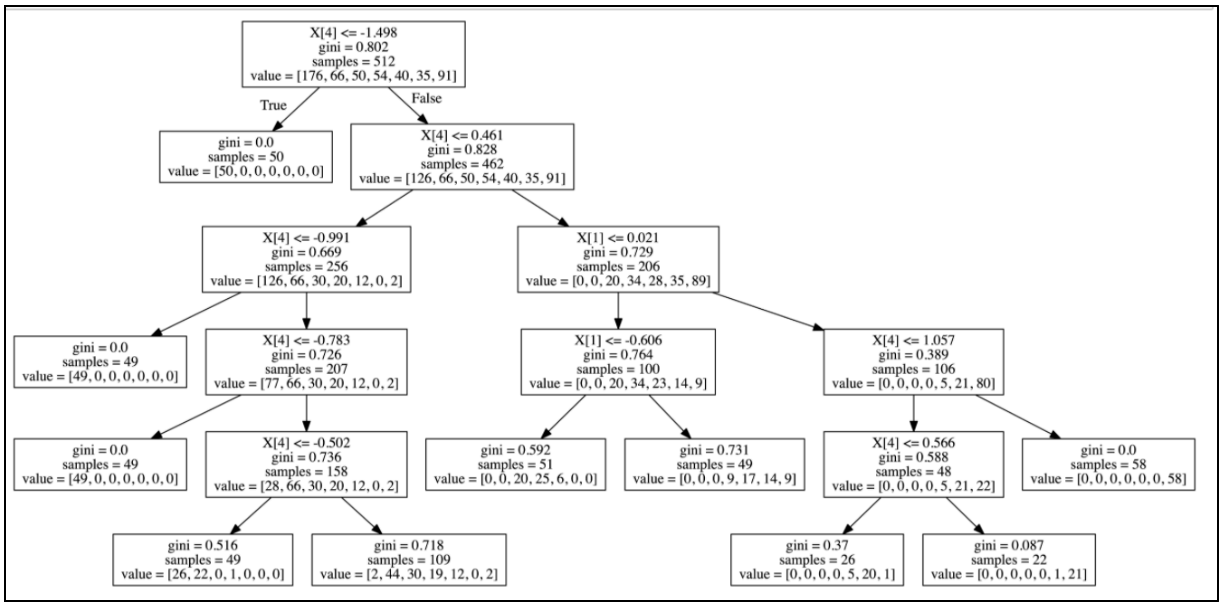
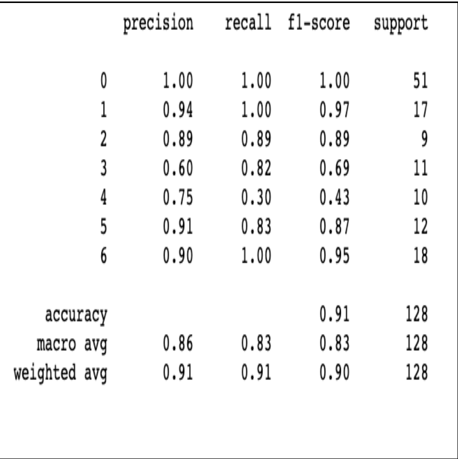

X[4] <= -1.498 gini = 0.802 samples = 512 value = [176, 66, 50, 54, 40, 35, 91]
True
False
gini = 0.0 samples = 50 value = [50, 0, 0, 0, 0, 0, 0]
X[4] <= 0.461 gini = 0.828 samples = 462 value = [126, 66, 50, 54, 40, 35, 91]
X[4] <= -0.991 gini = 0.669 samples = 256 value = [126, 66, 30, 20, 12, 0, 2]
X[1] <= 0.021 gini = 0.729 samples = 206 value = [0, 0, 20, 34, 28, 35, 89]
gini = 0.0 samples = 49 value = [49, 0, 0, 0, 0, 0, 0]
X[4] <= -0.783 gini = 0.726 samples = 207 value = [77, 66, 30, 20, 12, 0, 2]
X[1] <= -0.606 gini = 0.764 samples = 100 value = [0, 0, 20, 34, 23, 14, 9]
X[4] <= 1.057 gini = 0.389 samples = 106 value = [0, 0, 0, 0, 5, 21, 80]
gini = 0.0 samples = 49 value = [49, 0, 0, 0, 0, 0, 0]
X[4] <= -0.502 gini = 0.736 samples = 158 value = [28, 66, 30, 20, 12, 0, 2]
gini = 0.592 samples = 51 value = [0, 0, 20, 25, 6, 0, 0]
gini = 0.731 samples = 49 value = [0, 0, 0, 9, 17, 14, 9]
X[4] <= 0.566 gini = 0.588 samples = 48 value = [0, 0, 0, 0, 5, 21, 22]
gini = 0.0 samples = 58 value = [0, 0, 0, 0, 0, 0, 58]
gini = 0.516 samples = 49 value = [26, 22, 0, 1, 0, 0, 0]
gini = 0.718 samples = 109 value = [2, 44, 30, 19, 12, 0, 2]
gini = 0.37 samples = 26 value = [0, 0, 0, 0, 5, 20, 1]
gini = 0.087 samples = 22 value = [0, 0, 0, 0, 0, 1, 21]

Fig 11: Decision tree with optimal depth of 6 using cost complexity pruning.

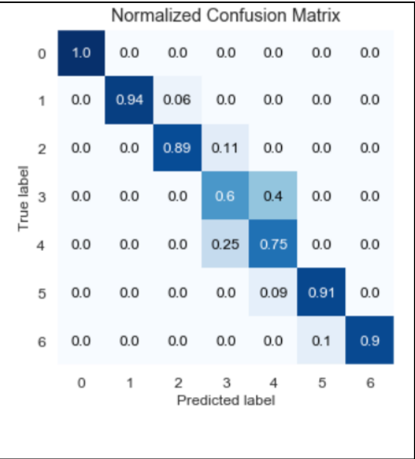
Having applied the cost complexity, the subtree with large cost complexity is chosen, the tuning parameter alpha (α) is chosen by using cross validation as 0.0229, with a depth of the tree as 6.



```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        51
           1       0.94      1.00      0.97        17
           2       0.89      0.89      0.89         9
           3       0.60      0.82      0.69        11
           4       0.75      0.30      0.43        10
           5       0.91      0.83      0.87        12
           6       0.90      1.00      0.95        18

    accuracy                           0.91       128
   macro avg       0.86      0.83      0.83       128
weighted avg       0.91      0.91      0.90       128
```



	0	1	2	3	4	5	6
0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.94	0.06	0.0	0.0	0.0	0.0
2	0.0	0.0	0.89	0.11	0.0	0.0	0.0
3	0.0	0.0	0.0	0.6	0.4	0.0	0.0
4	0.0	0.0	0.0	0.25	0.75	0.0	0.0
5	0.0	0.0	0.0	0.0	0.09	0.91	0.0
6	0.0	0.0	0.0	0.0	0.0	0.1	0.9

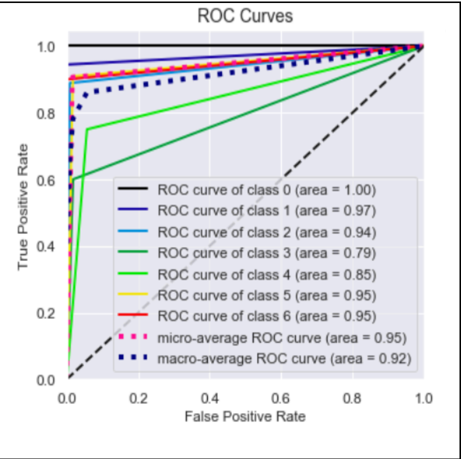
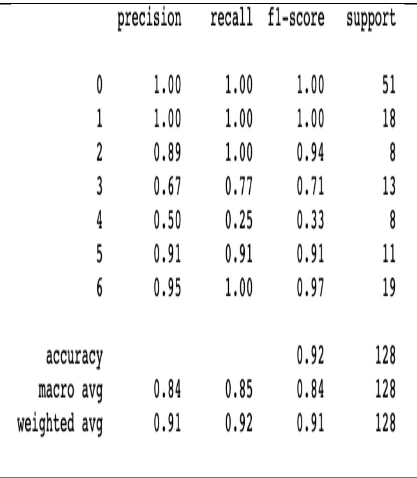


Fig 12: Performance evaluation of Decision Tree

Model 6: BAGGING

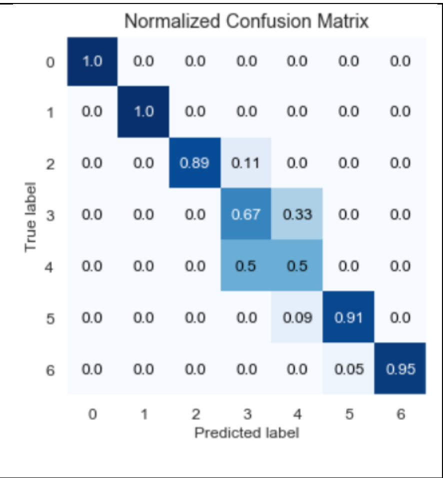
As Decision Tree performed well, I furthermore tried to improve the accuracy by using **bagging** to decision tree, in bagging all the predictors are bootstrapped the data the training samples with the replacement. Bagging with the base estimator as decision tree classifier and 100 estimators, gives an highest test accuracy of 92%, and training accuracy of 100%.



```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        51
           1       1.00      1.00      1.00        18
           2       0.89      1.00      0.94         8
           3       0.67      0.77      0.71        13
           4       0.50      0.25      0.33         8
           5       0.91      0.91      0.91        11
           6       0.95      1.00      0.97        19

    accuracy                           0.92       128
   macro avg       0.84      0.85      0.84       128
weighted avg       0.91      0.92      0.91       128
```



	0	1	2	3	4	5	6
0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	1.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.89	0.11	0.0	0.0	0.0
3	0.0	0.0	0.0	0.67	0.33	0.0	0.0
4	0.0	0.0	0.0	0.5	0.5	0.0	0.0
5	0.0	0.0	0.0	0.0	0.09	0.91	0.0
6	0.0	0.0	0.0	0.0	0.0	0.05	0.95

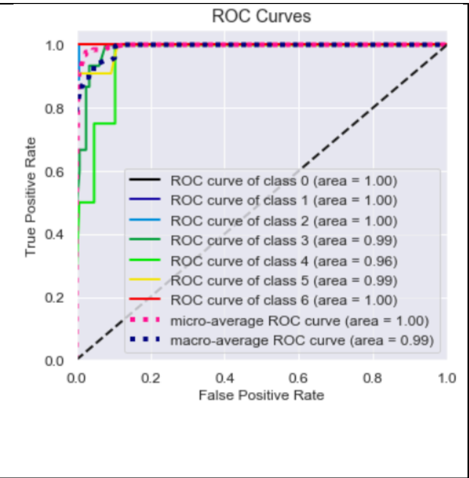
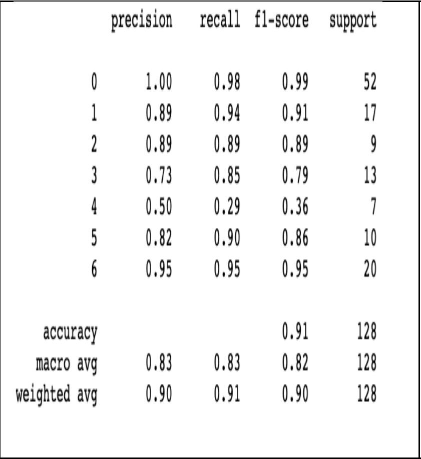


Fig 13: Performance evaluation of Bagging

Model 7: RANDOM FOREST

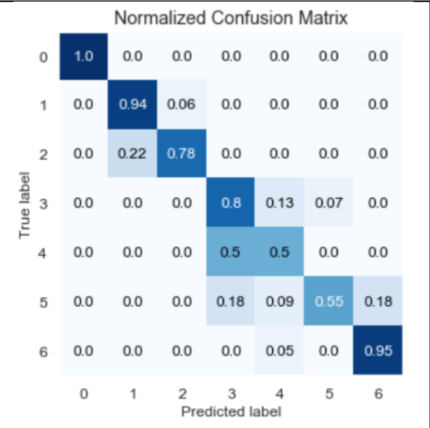
The **Random Forest model** is applied to the model, the model with n_estimators are chosen using cross validation as 60 also alpha of 0.06 from CV, and criteria of gini index while measuring the sample purity, the training accuracy of the model is 92.5% but the test accuracy is less, 91%, we can infer that the random forest model is does not fit well for this model. 4 classes are classified well with more than 90% accuracy.



```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99        52
           1       0.89      0.94      0.91        17
           2       0.89      0.89      0.89         9
           3       0.73      0.85      0.79        13
           4       0.50      0.29      0.36         7
           5       0.82      0.90      0.86        10
           6       0.95      0.95      0.95        20

    accuracy                           0.91       128
   macro avg       0.83      0.83      0.82       128
weighted avg       0.90      0.91      0.90       128
```



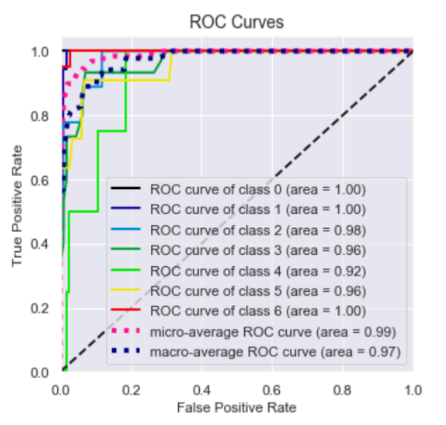
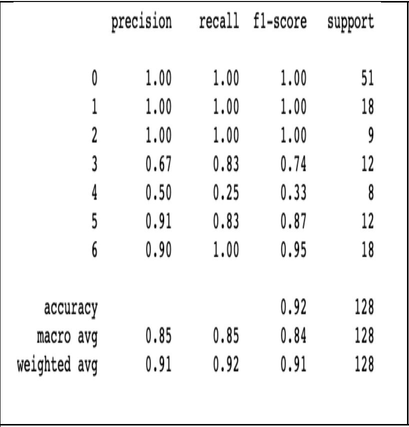


Fig 14: Performance evaluation of Random Forest

Model 8: BOOSTING

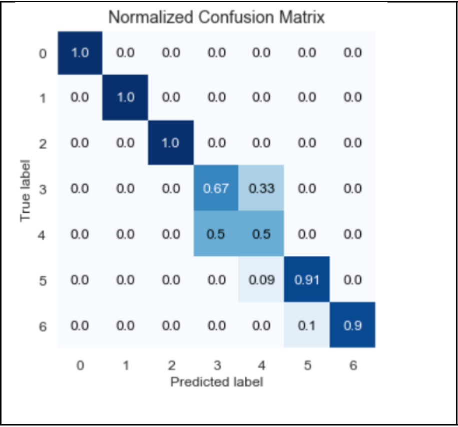
The most powerful technique of all in the decision tree models is **Boosting**, In boosting trees learns from previously built trees, it have 3 hyperparameters, these are number of trees(B), shrinkage parameter, for learning rate, and the number of splits at each tree. The hyper parameters are tuned using cross validation and set as B=100, Learning rate = 0.1, Split= 3, the training accuracy is 100% and the test is 92%, it classifies very well on 5 classes out of 7.



```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        51
           1       1.00      1.00      1.00        18
           2       1.00      1.00      1.00         9
           3       0.67      0.83      0.74        12
           4       0.50      0.25      0.33         8
           5       0.91      0.83      0.87        12
           6       0.90      1.00      0.95        18

    accuracy                           0.92       128
   macro avg       0.85      0.85      0.84       128
weighted avg       0.91      0.92      0.91       128
```



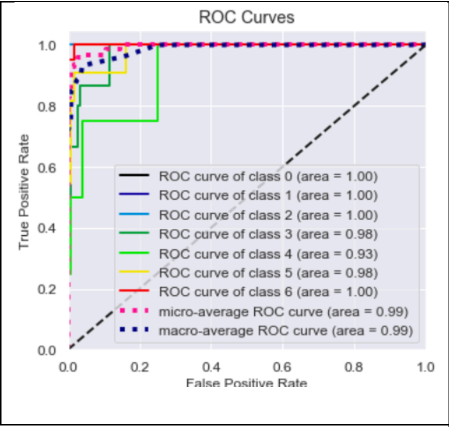
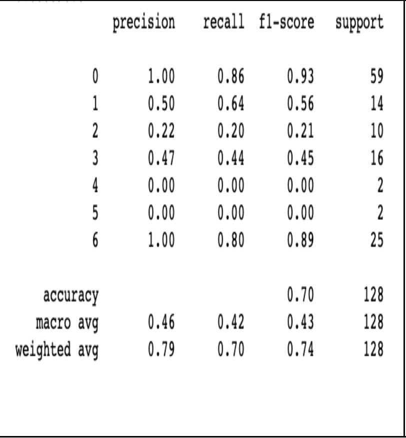


Fig 16: Performance evaluation of Bagging

Model 9: SVM

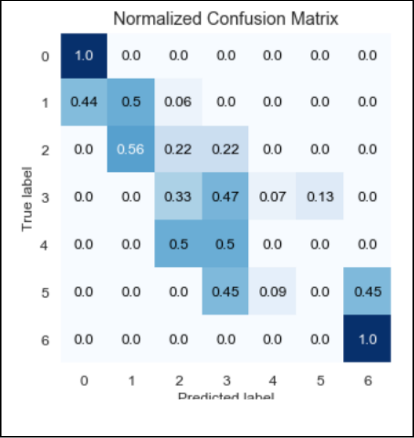
The non- linear classification model, **Support Vector Machine models** is used to fit the model, the SVM works with the kernals, surprisingly the non linear kernel Radial based function kernel gave a poor performance, with an accuracy of 65%, the model overfit the data, using the polynomial kernel the accuracy is 68%, the better accuracy of all the three kernels is by linear kernel the accuracy is 70%. Only class 0 and class 6 fit the data, Therefore, SVM are not the appropriate models for our dataset.



```
              precision    recall  f1-score   support

           0       1.00      0.86      0.93        59
           1       0.50      0.64      0.56        14
           2       0.22      0.20      0.21        10
           3       0.47      0.44      0.45        16
           4       0.00      0.00      0.00         2
           5       0.00      0.00      0.00         2
           6       1.00      0.80      0.89        25

    accuracy                           0.70       128
   macro avg       0.46      0.42      0.43       128
weighted avg       0.79      0.70      0.74       128
```



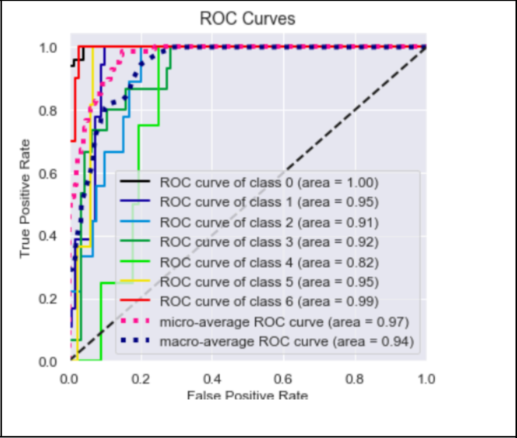


Fig 17: Performance evaluation of SVM

Model 10: PCA+CLASSIFICATION

This model can be considered as a part visualization, Performed **Principle component analyses**, to reduce the dimension of feature space, By observing the loading vector of reduced 2 dimensional PCA model, we can say that the 1st principal component explains the inputs 1 and 2 Node Number and Thread Number, and the 2nd principal component explain the last feature T/R ratio. From this we can say that the 'Node Number' and 'Thread Number' has highest variation. And perpendicularly the 2nd highest variance is for T/R. The First principle component corresponds to the features node number and Thread Number and the second principle component corresponds to T/R.

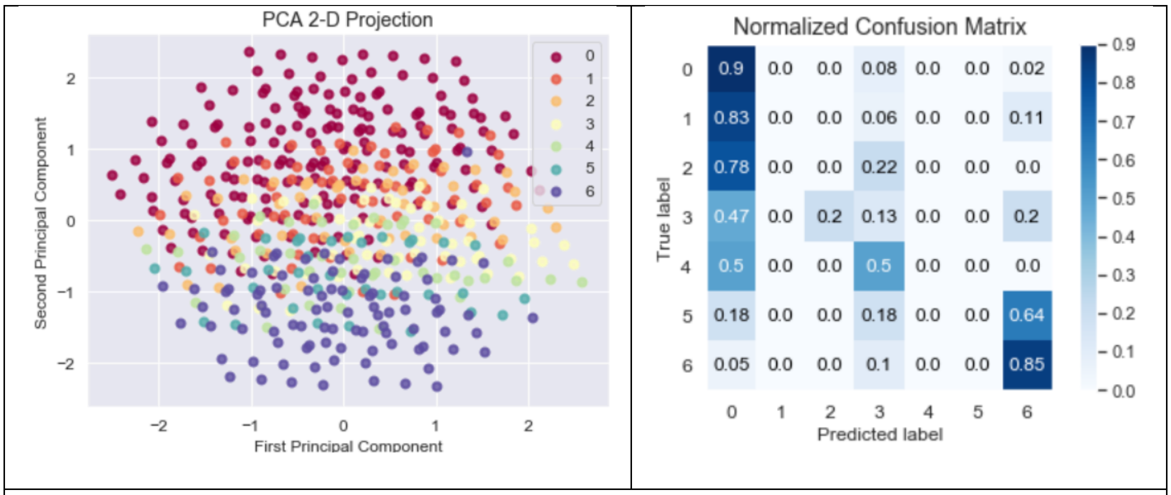


Fig 18: 2D projection of PCA, Confusion matrix Model after applying PCA

Observations from the 2 Dimensional projection: All the data points with classes (0-6) are spread all over first component(Node number, Thread Number)but data with class 0 have high T/R value above 0 in 2nd Principle component, the Class 6 but have less T/R ratio. Rest of all classes has average T/R ratio.

The PCA applied data is fit into the SVM model gives lowest train and test accuracy of 51%. Therefore, PCA+ classification is not the best model to classify Channel Waiting Time.

VII Feature Set

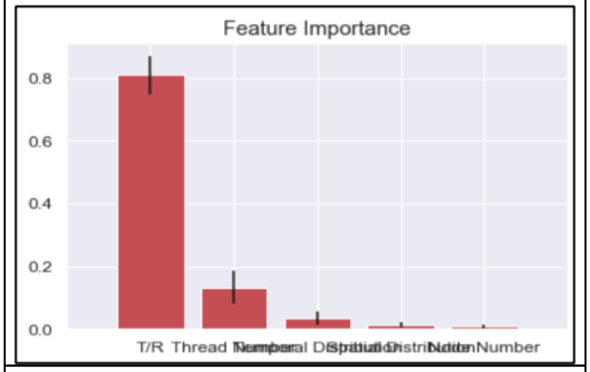


Fig: Feature Importance after fitting model on random Forest

From decision tree models it is possible to know the importance of features, I have performed feature importance using a function plot_feature_importances from scikitplot API reference. Observing feature Importance graph only 2 features more important, i.e., T/R ratio and Thread Number, it is same as the result from subset selection

VIII Performance and Summary of Report:

- The Data is visualised using various methods including PCA and found outliers in the data and the data is cleaned.
- The inputs Spatial Distribution and Temporal Distribution are encoded. The output variable is labelled to 7 classes using cut-points to make it classify
- Features are selected after performing Subset selection method, Forward Stepwise selection method, regularizing using Lasso and Ridge regression, The most important Features selected are 'Thread Number' and 'T/R
- Splitted the dataset into training (80%) and test data (20%) for validation purpose
- Performed Logistic Regression and gained an accuracy of 75% which hints at the possibility of non-linear decision boundary
- The data is fitted on LDA, which gave an accuracy of 72% which again denotes the absence of linearity
- The non-linear classification model (QDA) is applied to data, the accuracy is 84%, the data distribution might be gaussian with different variance for all the classes
- After performing KNN on data the with the tuned hyperparameter K=10 using cross validation, the accuracy is 70%, as a non-parametric model it requires more data to fit the model, the problem is data deficiency
- The decision trees model is applied to data, with the pruning, reached an accuracy of 91%
- Performed bagging with 100 estimators, the accuracy is 92%
- Random Forest is applied to data, with estimators 500, and learning rate 0.1 the accuracy is 91%, this model overfit the data.
- Performed Boosting with tuned parameters of 100 estimators, learning rate 0.1 and d=3, gave highest accuracy of 92%

- Using support vector machines, with linear kernel the accuracy is 78%, the polynomial and radial kernel overfits the data
- Performed PCA, reduced dimensions from 5 to 2 and fitted the components using SVC to achieve an accuracy 51% which is lowest accuracy. The poor performance is due to the low ratio of feature and data size
- Applied all these models for reduced feature set and also to other 2 output variables
- The best performance is achieved with Boosting which gave an accuracy of 92% when channel waiting time is target and all features are used. The best accuracy of 89% is achieved with boosting using 2 features and accuracy of 88% using Process utilization as target.

XI Discussion of Results:

On observing the performance of the models, the data is spread non linearly. The best performance is given by Boosting after tuning the hyperparameters for all of the outputs in the model. The table below shows the performance of models as a function of accuracy.

	Channel Waiting Time (with all Features)	**Channel Waiting Time (with 2 best Features from feature extraction)**	**Processor Utilization**	**Network Response time**
Logistic Regression	0.75	0.66	0.57	0.70
LDA	0.73	0.78	0.56	0.75
QDA	0.84	0.82	0.56	0.78
KNN K=5	0.67	0.80	0.74	0.81
K=10	0.70	0.89	0.78	0.75
K=15	0.65	0.82	0.73	0.78
Decision Tree	0.91	0.89	0.84	0.81
Bagging	0.92	0.89	0.57	0.70
Random Forest	0.91	0.89	0.87	0.78

Boosting	0.92	0.89	0.88	0.87
SVM	0.78 (Linear)	0.72	70 (RBF)	0.77 (RBF) 0.76 (Linear)
PCA	0.51	0.57		0.48

X Problems Faced and Conclusion

The number of data points and features are very less for this dataset, The non-parametric models and other models could not train the data efficiently.

The Optical Interconnection Data with all the output classifiers fit into the non-linear models gives the high accuracy, we cannot say one model is the best model for the data. After feature selection all the non-linear models gained same performance of accuracy 89%, boosting is the best model out of all machine learning models for the optical interconnection data.

References:

[1] M.F. Akay, CÂ¸.I. Aci, F. Abut, Predicting the Performance Measures of a 2-Dimensional Message Passing Multiprocessor Architecture by Using Machine Learning Methods. Neural Network World 71(5):1907-1931. DOI: 10.14311/NNW.2015.25.013.

[2] AcÄ±, Ã‡.Ä°. & Akay, M.F. A hybrid congestion control algorithm for broadcast-based architectures with multiple input queues. J Supercomput (2015) 71: 1907.

[3] Sicotte, B. (2018, June 11). Xavier Bourret Sicotte. Retrieved May 5, 2020, from https://xavierbourretsicotte.github.io/subset_selection.html

[4] https://scikit-learn.org/stable/