2/13/2020

Classifying a GPU run process as high or low time consuming using various classification Algorithms.

SARTHAK MOHAPATRA
THE UNIVERSITY OF TEXAS AT DALLAS, US

INTRODUCTION –

In this project, we will learn and implement SVM, Decision Trees and XGBoost Algorithms method. The purpose of this project is to compare different classification algorithms and test its capabilities for classifying records in a dataset. The dataset contains information about GPU kernel performance on matrix multiplication (A * B = C) where, A, B and C are metrices. Our goals are highlighted below:

- Implementing SVM, Decision Trees and XGBoost Algorithms to classify the GPU run time as high or low time-consuming process.

In this project, we have performed various experimentations by using different kernels with SVM algorithm, pruning Decision Trees and XGBoost to avoid overfitting. Based on various performance measures, the algorithms were evaluated and the best algorithm for each dataset was finalized.

DATASET & FEATURE DESCRIPTION –

For this project, we have used two datasets to fully understand the predictive capabilities of the algorithms on various type of datasets:

- For this project, we have used the SGEMM GPU kernel performance Data Set available for download at UCI ML Website. The dataset measures the running time of various matrix multiplication processes where each matrix is of shape 2048 × 2048. The total number of observations in the dataset is 241600. For each test, four runs were performed, and their results were presented in the file. For our project, we have taken the average of four runs and have considered it as our target/dependent variable.

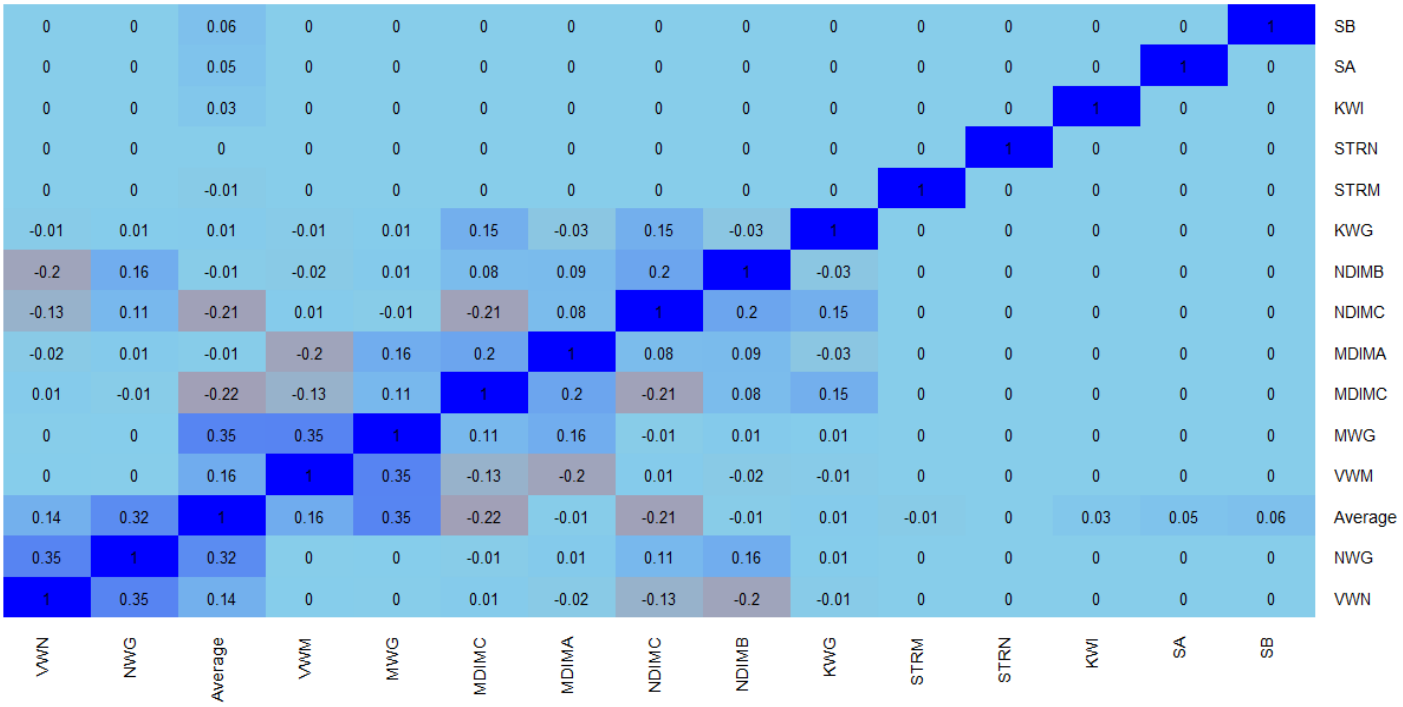
The datasets were divided into training (70%) and validation (30%) dataset with 70:30 split ratio.

DESCRIPTIVE STATISTICS AND EXPLORATORY DATA ANALYSIS -

Since we are trying to accurately predict the GPU run time and correctly classify a run time as high or low time consuming, the features that we are going to use will play a major role in the effectiveness of the model while predicting or classifying various runs. So, to understand every variable and how their presence or their correlation impacts the run time, descriptive statistics and exploratory data analysis was performed.

Correlation Among Variables -

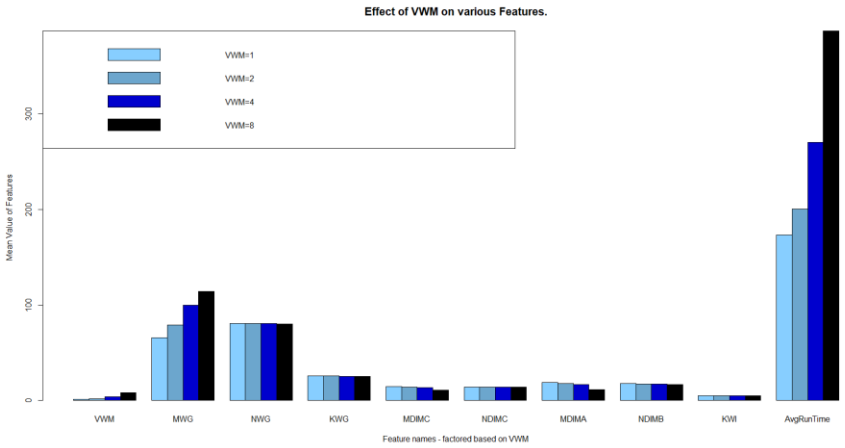
To understand the correlation among variables and how their presence in the model is going to affect other independent variables or features, we have plotted the below heat-map:



It explains the correlation among variables in the dataset. The key takeaways from the above correlation matrix/ heat-map is mentioned below:

- The dependent variable Average has the positive correlation with MWG and NWG. But the correlation coefficient is not high.
- The dependent variable Average has negative correlation with NDIMC and MDIMC. But the correlation coefficient is not high.
- The dependent variables have highest positive correlation of 0.35 with MWG and highest negative correlation of -0.22 MDIMC.
- MWG has a small positive correlation with VWM. Also, NWG has a small positive correlation with VWN.
- MDIMC is having a small negative correlation with NDIMC.

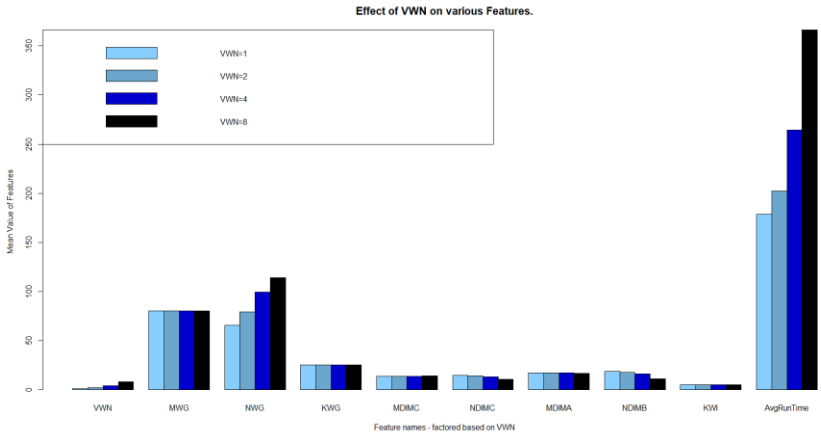
For the correlated variables, the correlation coefficient is small enough and are not going to impact much. To further analyze the impact of independent variables on the dependent variable (Average Run Time), the following graphs was plotted:



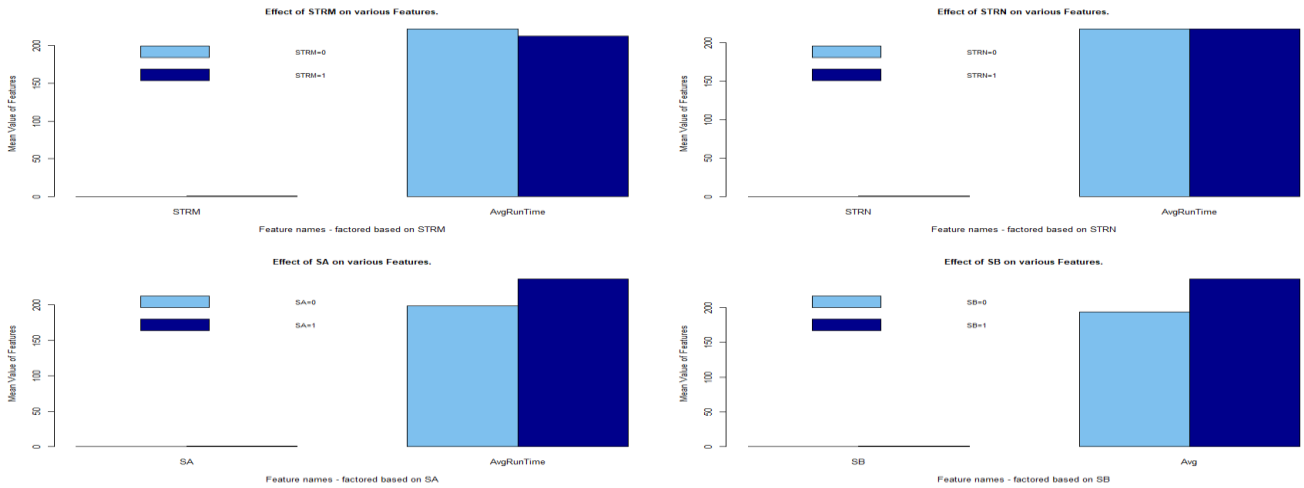
The graph on the left explains us the effect of VWM on Average Run Time and all other ordinal variables/ features. From the graph we can observe that with increase in the order of VWM, the Average Run Time has increased significantly. Similarly, with increase in order of VWM, the order of MWG also seems to increase signifying the records having with higher order of VWM has higher order value of MWG. For all other variables/ features, there is not much significant impact of VWM as we can see that the changes are very minimal or unchanged.

To understand the effect of VWN on other features and dependent variable, let's analyze the below graph:

From the graph on the right, we can observe the effect of VWN on Average Run Time and all other ordinal variables/ features. It can be clearly seen that with increase in the order of VWN, the Average Run Time has increased significantly. It also explains the highest positive correlation of VWN and Average Run time in the Heat-Map. Similarly, with increase in order of VWN, the order of NWG also seems to increase signifying the records having with higher order of VWN, also has higher order value of NWG. For all other variables/ features, there is not much significant impact of VWN as we can see that the changes are very minimal or unchanged.



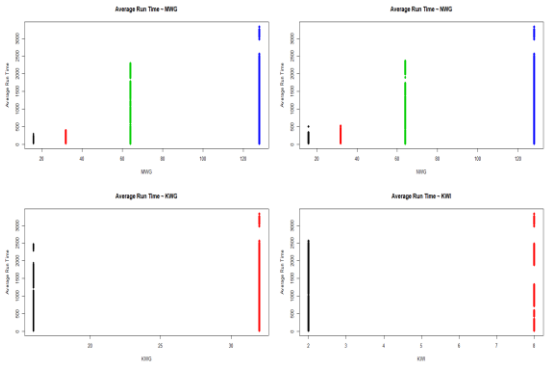
Since the categorical variables STRM, STRN, SA and SB are not having correlation with the ordinal variables, let's analyze the graph below to understand the effect of the categorical variables on the Average Run Time:



(Please note that since the value of the categorical variables are very small compared to Avg Run Time, the bars for the categorical variables are not clear)

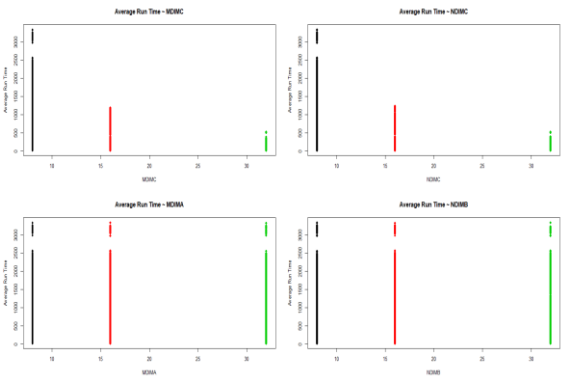
From the above graph, we can see that for STRM and STRN, there is not much variation in the Average Run Time. As compared to a run where the stride is not enabled to access off-chip memory in single thread (STRM = 0), the Average Run Time slightly decreases when stride is enabled to access off-chip memory in single thread (STRM = 1). Unlink STRM/ STRN, as compared to a run where per-matrix manual caching of the 2D workgroup tile is not there(SA=0/ SA=1), the Average Run Time increases when there is per-matrix manual caching of the 2D workgroup tile(SA=1/ SB=1) in a run process. Also, we can see that STRN has no impact on Average Run Time which is constant and has same mean value.

To understand the effect to the other ordinal features on Average Run Time, the below graphs explains the changes in the Average Run Time with variation in the order of the features:



In the figure to the right, we can observe that there are a greater number of observations in the dataset with higher order of the MWG & NWG. Also, with increase in order of MWG & NWG, there are good number of observations where the Average Run Time is very high. For KWG and KWI, although there are few records with high Average Run Time, but overall there is not much impact of KWI and KWG.

In the figure to the right, we can see that MDIMA and NDIMB is not having impact on the Average Run Time. We have similar number of observations in each category MDIMA and NDIMB and the variation in records are almost same. But, for MDIMC and NDIMC, we can see that as the order of the MDIMC and NDIMC increases, the Average Run Time decreases very significantly, and we have very few transactions which have higher order value of MDIMC and NDIMC.



Data Preparation –

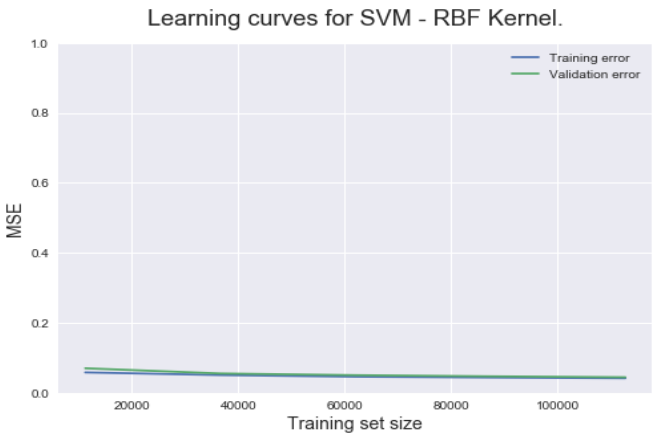
All the features in the dataset contains data in different ranges. Since we are going to use the Gradient Descent method to perform Linear Regression and Logistic Regression method, all the features in the dataset were scaled using the mean normalization method. Following the normalization process, each feature has mean 0 and standard deviation as 1.

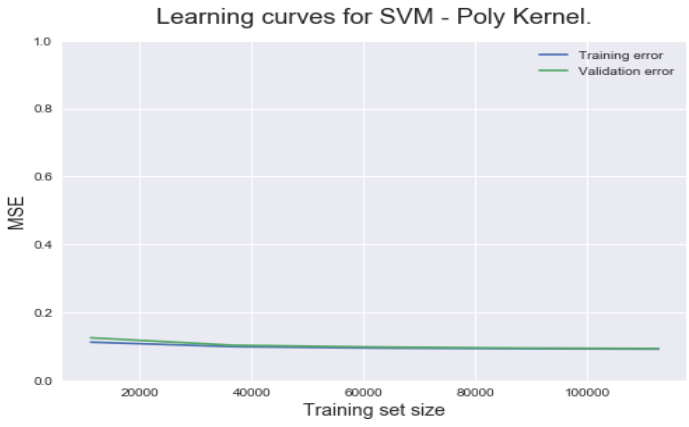
For Logistic Regression, the median of the Average Run Time was taken and observations below median were changed to 0 (Low run) and above median to 1 (High run) to convert into a binary class problem.

CLASSIFICATION MODELS USING SUPPORT VECTOR MACHINES –

The SVM Algorithm with Linear, Polynomial and RBF kernel was used to perform classification of records in both the dataset. The below snapshots highlight the error curves for both training and validation data set for the Polynomial and Redial kernel of the SVM algorithm:

GPU Performance Datasets -

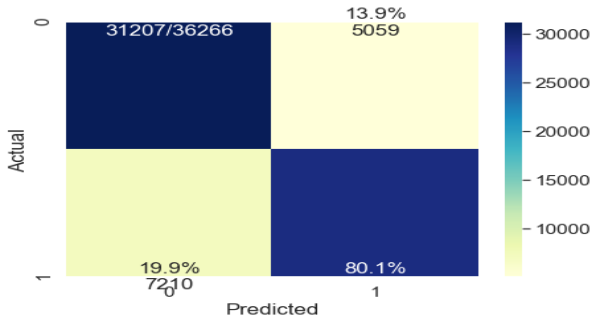




The figure of the left showcases the error rate change for the Radial kernel in SVM algorithm. We can see that the initial error rate in the validation data set was higher and eventually decreased to meet the minimum. Similar results were obtained with the Polynomial kernel too signifying that both the kernels are equally good in predicting the class of the record in the GPU performance dataset.

1- Experimenting with LINEAR KERNEL function in Support Vector Machine with both the datasets -

GPU Runtime Classification Dataset –



Report SVM Linear Kernel

	precision	recall	f1-score	support
0	0.81	0.86	0.84	36266
1	0.85	0.80	0.83	36214
accuracy			0.83	72480
macro avg	0.83	0.83	0.83	72480
weighted avg	0.83	0.83	0.83	72480

With the Linear Kernel function, the SVM model's performance on the GPU validation dataset is:

Key Points:

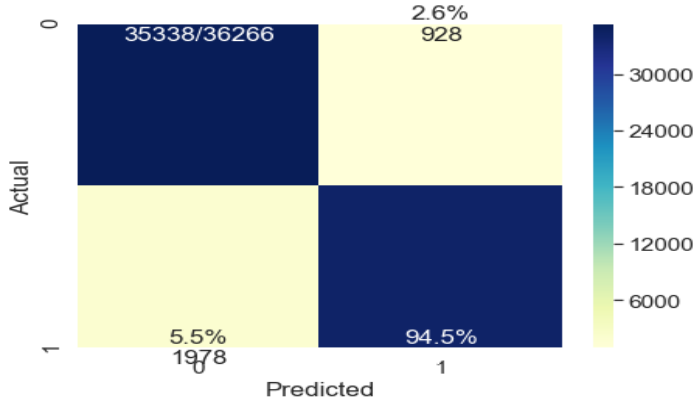
- The accuracy of the model on the validation dataset is ~83 %.
- The Precision of the model on the validation dataset for class 0 is ~81%.
- The Precision of the model on the validation dataset for class 1 is ~85%.

With the Linear Kernel function, the model is good enough in classifying ~86% of the 0 class records correctly and ~80% of the 1 class records correctly as specified in the recall column of the report.

2- Experimenting with RBF KERNEL function in Support Vector Machine with both the datasets –

GPU Run Time Classification Data –

With the RBF Kernel function, the SVM model's performance on the GPU validation dataset is highlighted below:



Report SVM RBF Kernel

	precision	recall	f1-score	support
0	0.95	0.97	0.96	36266
1	0.97	0.95	0.96	36214
accuracy			0.96	72480
macro avg	0.96	0.96	0.96	72480
weighted avg	0.96	0.96	0.96	72480

Key Points:

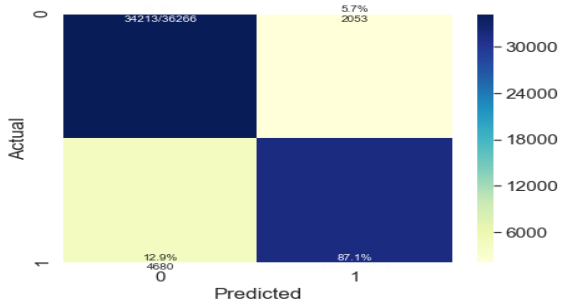
- The accuracy of the model on the validation dataset is ~96 %.
- The Precision of the model on the validation dataset for class 0 is ~95%.
- The Precision of the model on the validation dataset for class 1 is ~97%.

With the RBF Kernel function, the model does an excellent job in classifying ~97% of the 0 class records correctly and ~95% of the 1 class records correctly as specified in the recall column of the report.

3- Experimenting with Polynomial KERNEL function in Support Vector Machine on both datasets –

GPU Run Time Classification Data –

With the Polynomial Kernel function, the SVM model's performance on the GPU validation dataset is highlighted below:



Report GPU Poly Kernel SVM

	precision	recall	f1-score	support
0	0.88	0.94	0.91	36266
1	0.94	0.87	0.90	36214
accuracy			0.91	72480
macro avg	0.91	0.91	0.91	72480
weighted avg	0.91	0.91	0.91	72480

Key Points:

- The accuracy of the model on the validation dataset is ~91 %.
- The Precision of the model on the validation dataset for class 0 is ~88%.
- The Precision of the model on the validation dataset for class 1 is ~94%.

With the Polynomial Kernel function, the model does an excellent job in classifying ~94% of the 0 class records correctly and ~87% of the 1 class records correctly as specified in the recall column of the report.

CLASSIFICATION MODELS USING DECISION TREE ALGORITHM –

The decision tree algorithm was used on both the datasets for classifying the records in the respective dataset correctly.

The Gini Index metric was selected as the criteria in Decision Tree splitting because in both the datasets, the number of features and the records were high. The Gini Index is calculated by subtracting the sum of the squared probabilities of each class from one and hence it favors larger partitions. So, for our experimentation, the Gini Index metric was chosen as the split criteria.

The below snapshots show the error curve with the fully grown Decision Tree for both the datasets:

GPU Performance Dataset

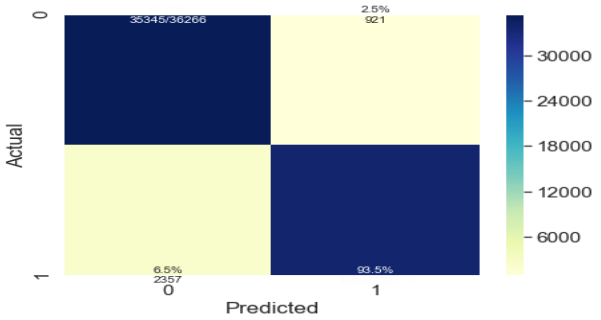
The above figure on left shows the error curve for both training and validation dataset for the GPU performance dataset. We can observe that for the validation dataset, the error rate was higher than the test dataset and decreased gradually. The higher error rate in the validation dataset is an indication of no overfitting.

The above figure on the right shows the error curve for both training and validation dataset for the customer churn dataset. We can observe that in case of the validation dataset, the error rate is significantly higher than the training dataset.

Pruning Experimentation on GPU Run Time Classification Data –

The below snapshots show the Accuracy Reports and Confusion Matrix for trees with various depths:

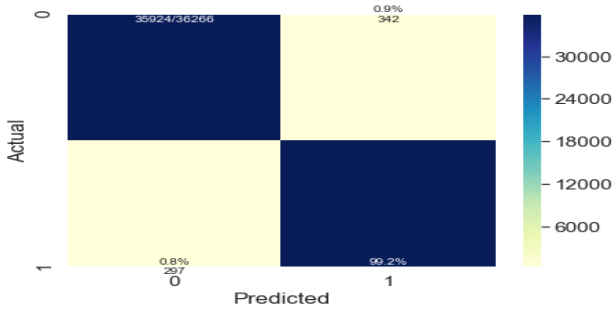
Max_Depth = 10



Report - GPU - Tree Depth 10

	precision	recall	f1-score	support
0	0.94	0.97	0.96	36266
1	0.97	0.93	0.95	36214
accuracy			0.95	72480
macro avg	0.96	0.95	0.95	72480
weighted avg	0.96	0.95	0.95	72480

Max_Depth = 20

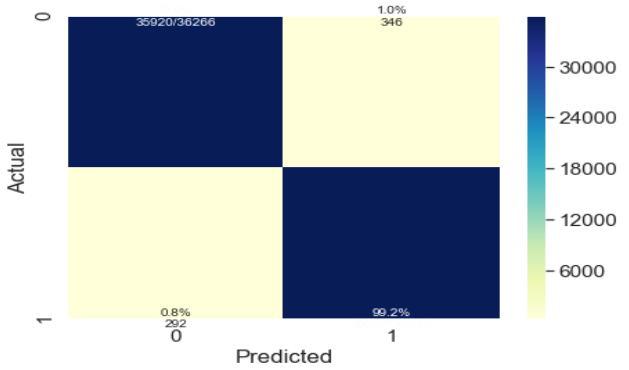


Report Tree GPU Depth 20

	precision	recall	f1-score	support
0	0.99	0.99	0.99	36266
1	0.99	0.99	0.99	36214
accuracy			0.99	72480
macro avg	0.99	0.99	0.99	72480
weighted avg	0.99	0.99	0.99	72480

From the above reports we can observe that at maximum depth 20, the decision tree model is able to classify records with an accuracy of ~99%. For both the classes, precision and the sensitivity value are close to ~99%. For the model with maximum depth 10, the decision tree model can classify records with an accuracy of ~95%. For the class 1 records, the sensitivity is ~93% and for class 0, it's ~97%.

The below snapshots highlight the accuracy reports and confusion matrix for the fully grown tree:



Report - GPU - Tree Max Depth

	precision	recall	f1-score	support
0	0.99	0.99	0.99	36266
1	0.99	0.99	0.99	36214
accuracy			0.99	72480
macro avg	0.99	0.99	0.99	72480
weighted avg	0.99	0.99	0.99	72480

From the above report we can observe that the various measures for the fully grown tree and the tree with depth 20 is almost identical. Hence, as compared to the fully grown tree, we can opt for the tree with max depth 20. Similarly, as compared to the fully grown tree, the tree with max depth 10 performs at par in classifying records for both classes. Hence, opting for the tree with Max Depth 10 will be the best option as we can avoid over fitting with lesser number of splits.

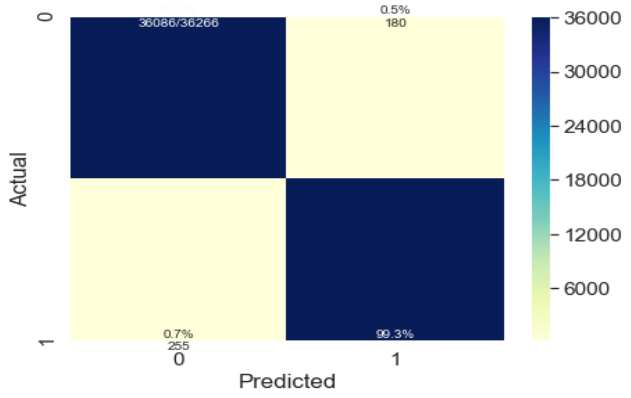
CLASSIFICATION MODELS USING XGBOOST ALGORITHM –

The XGBoost algorithm was used on both the datasets for classifying the records in the respective dataset correctly. The below snapshots show the error curve for XGBoost algorithm with depth 5:

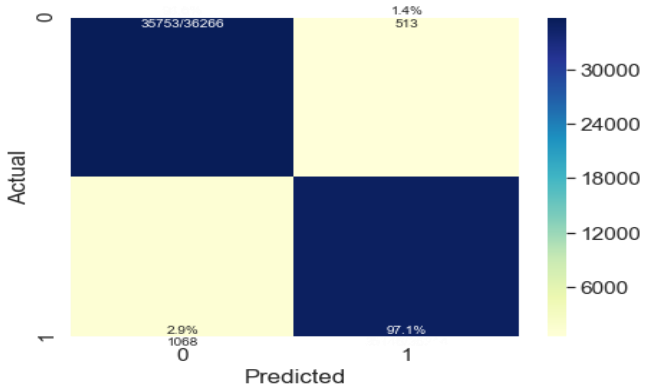
The figure on left shows the error curve for both training and validation dataset for the GPU performance dataset. We can observe that for the validation dataset, the error rate was higher in the test dataset and decreased gradually. The higher error rate in the validation dataset is an indication of no overfitting.

Pruning Experimentation on GPU Run Time Classification Data –

Depth – 5



Depth – 10



Report GPU XGBoost - Depth 5

	precision	recall	f1-score	support
0	0.97	0.99	0.98	36266
1	0.99	0.97	0.98	36214
accuracy			0.98	72480
macro avg	0.98	0.98	0.98	72480
weighted avg	0.98	0.98	0.98	72480

Report GPU XGBoost - Depth 10.

	precision	recall	f1-score	support
0	0.99	1.00	0.99	36266
1	1.00	0.99	0.99	36214
accuracy			0.99	72480
macro avg	0.99	0.99	0.99	72480
weighted avg	0.99	0.99	0.99	72480

From the above reports and confusion matrices, we can observe that at maximum depth 10, the XGBOOST model is able to classify records with an accuracy of ~99%. For the class 1 records, the sensitivity is ~99% and for class 0, it's 100%.

For the figures on the left, we can observe that with a maximum depth of 5, the accuracy of the model is ~98%. Similarly, the precision and sensitivity of the model for both the classes is close to ~98%.

Hence, opting for the model with maximum depth of 5 will provide us the best results without overfitting.

CROSS-VALIDATION WITH ALL ALGORITHMS –

Cross validation was performed on the dataset with the 6-fold cross validation technique and the cross-validation scores for the dataset is mentioned below:

For the datasets, with the application of k-fold cross-validation technique, the cross-validation scores with all algorithms are almost the same.

	CV1	CV2	CV3	CV4	CV5	CV6
SVM-LIN	0.825759	0.827001	0.838079	0.834023	0.826641	0.831774
SVM-RBF	0.950170	0.949673	0.952649	0.949752	0.948257	0.952231
SVM-POLY	0.898187	0.899429	0.904967	0.902070	0.902558	0.900075
D-TREE Full	0.979224	0.980714	0.980877	0.979967	0.978806	0.982035
D-TREE D-10	0.949839	0.953563	0.950579	0.953725	0.953059	0.954715
D-TREE D-20	0.889248	0.891234	0.891060	0.891887	0.892955	0.893038
XGB D-5	0.981872	0.983942	0.985017	0.982533	0.982366	0.984187
XGB D-10	0.976409	0.977320	0.979305	0.975331	0.976074	0.977564

COMPARISION OF ALGORITHMS FOR BOTH DATASETS –

The SVM with various kernel functions, Decision Tree and XGBoost algorithms was used to classify the records in the GPU Performance dataset. Various performance measurement parameters for the algorithms taken on the datasets are highlighted below:

	Accuracy	Precision	Recall
SVM-LIN	0.830726	0.800906	0.851481
SVM-RBF	0.959906	0.945380	0.973609
SVM-POLY	0.907105	0.870768	0.938875
D-TREE Full	0.991198	0.991937	0.990460
D-TREE D-10	0.954774	0.934915	0.973518
D-TREE D-20	0.991184	0.991799	0.990568
XGB D-5	0.993998	0.992959	0.995019
XGB D-10	0.978187	0.970509	0.985614

The table on the left shows the various performance measure for all the algorithms applied on the GPU Performance dataset. The XGBoost algorithm with max depth 5 performs best in classifying a GPU Run Time followed by Support Vector Machine algorithms with RBF kernel function.