SUMMARY

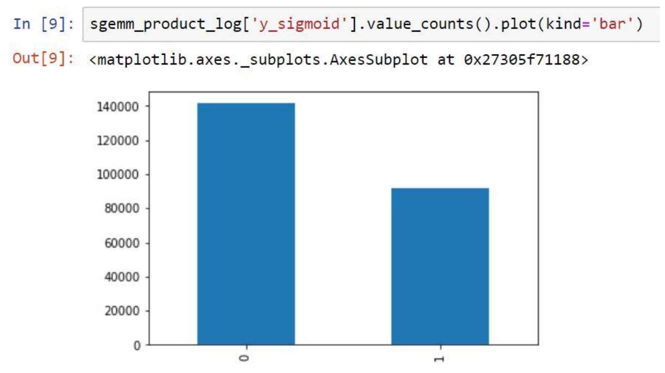
- Applying Support Vector Machine (SVM), Decision trees and Boosting on two data sets.
- Implementing and using cross-validation for both datasets for all the three algorithms.
- Obtaining train and test error rates for the algorithms for both the data sets.
- Comparisons of various performance metrics for algorithms for both the data sets.

ABOUT THE DATASETS

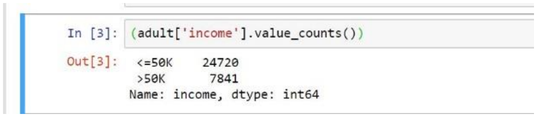
- The first dataset on which the algorithms are implemented is the SGM GPU kernel performance dataset which is available on the UCI Machine Learning Repository at https://archive.ics.uci.edu/ml/datasets/SGEMM+GPU+kernel+performance. The dataset consists of 241600 observations related to 14 variables. The dependent variable is the average of the performance times (in milliseconds) of the four independent runs 'Run1', 'Run2', 'Run3', 'Run4'.There is no missing value in any of the columns, and so missing value imputation is not required.
- The second dataset is the 'Adult income' dataset. It consists of various factors that influence an individual's income. The income is divided into two classes : <=50K and >50K. The dataset has a total of 48842 observations. It has a total of 14 independent variables and one dependent variable in the form of income categories. This dataset is available on the UCI Machine Learning Repository at https://archive.ics.uci.edu/ml/datasets/Adult . The dataset has sufficient number of features and observations which are distributed in a way so as to make a good model that generalizes well, and all three algorithms will work well with such a dataset. This is the reason why this dataset was chosen.

DATA PREPARATION

For both the datasets, data was cleaned and processed prior to applying cross validation and training. Descriptive statistics was performed on the dataset to find out the nature of distribution of various data points.



```
In [9]: sgemm_product_log['y_sigmoid'].value_counts().plot(kind='bar')
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x27305f71188>
```



```
In [3]: (adult['income'].value_counts())
Out[3]:  <=50K    24720
         >50K      7841
        Name: income, dtype: int64
```

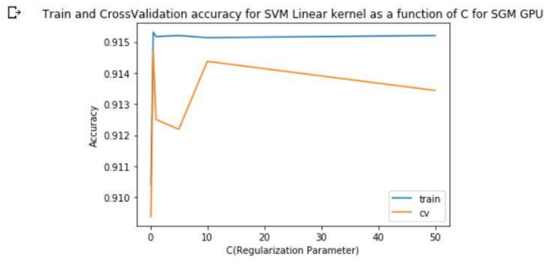
Since most of the variables in both the datasets were categorical, dummy variables were created. The standard deviations in the table indicate that the values are significant. The numeric education level and the nominal education level are highly correlated while other attributes are weakly correlated. The lower values of education level are found more in the <=50K category while the higher values are in preponderance in the >50K category. The independent features in both the datasets are scaled and transformed.

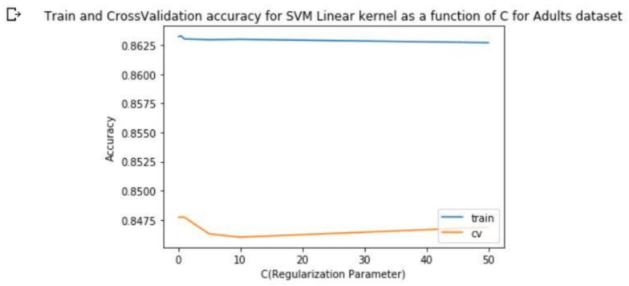
EXPERIMENTATION

The different algorithms were run on the datasets for which the observations are as under:

1. **Support Vector Machines**

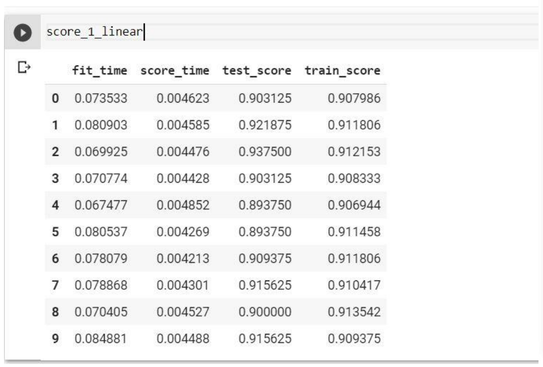
   The support vector machine algorithm was imported and implemented using three kernels : 'linear','rbf' and 'sigmoid' . The reasons for choosing these kernels over others is that linear and rbf kernels generalize well and are easier to tune. Also, rbf and sigmoid kernels work well with datasets that are not easily linearly separable, which is the case here. For each of these kernels, the different values of regularization parameter 'C' were experimented with. The scores for these different values were obtained. The error rates vs 'C' plot for both the datasets is as under:





There is a significant difference between the accuracies of training and validation data in the case of adults dataset.

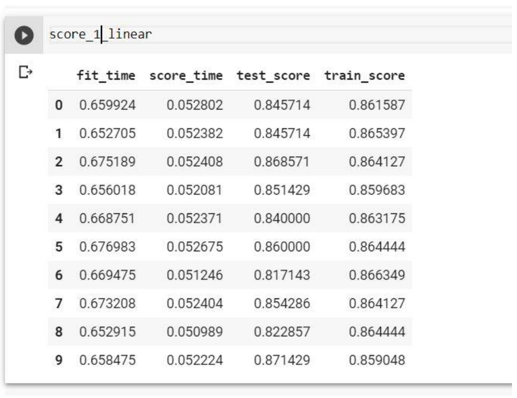
The clock times for linear kernel for the two datasets is as under:



```
score_1_linear
```

	fit_time	score_time	test_score	train_score
0	0.073533	0.004623	0.903125	0.907986
1	0.080903	0.004585	0.921875	0.911806
2	0.069925	0.004476	0.937500	0.912153
3	0.070774	0.004428	0.903125	0.908333
4	0.067477	0.004852	0.893750	0.906944
5	0.080537	0.004269	0.893750	0.911458
6	0.078079	0.004213	0.909375	0.911806
7	0.078868	0.004301	0.915625	0.910417
8	0.070405	0.004527	0.900000	0.913542
9	0.084881	0.004488	0.915625	0.909375

SGM GPU Dataset

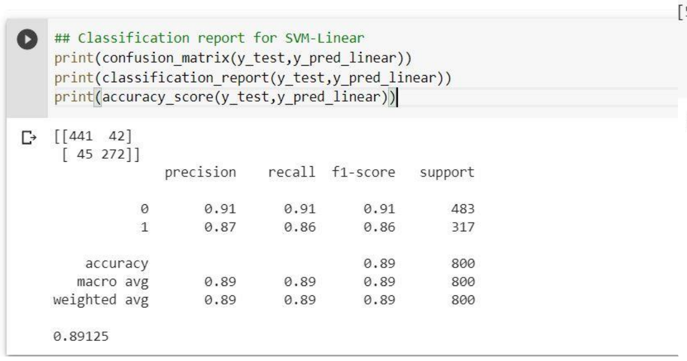


```
score_1_linear
```

	fit_time	score_time	test_score	train_score
0	0.659924	0.052802	0.845714	0.861587
1	0.652705	0.052382	0.845714	0.865397
2	0.675189	0.052408	0.868571	0.864127
3	0.656018	0.052081	0.851429	0.859683
4	0.668751	0.052371	0.840000	0.863175
5	0.676983	0.052675	0.860000	0.864444
6	0.669475	0.051246	0.817143	0.866349
7	0.673208	0.052404	0.854286	0.864127
8	0.652915	0.050989	0.822857	0.864444
9	0.658475	0.052224	0.871429	0.859048

Adult Dataset

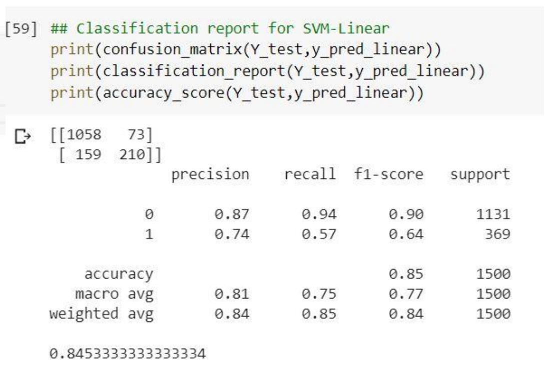
Classification reports:



```
## Classification report for SVM-Linear
print(confusion_matrix(y_test,y_pred_linear))
print(classification_report(y_test,y_pred_linear))
print(accuracy_score(y_test,y_pred_linear))
```

```
[[441  42]
 [ 45 272]]
              precision    recall  f1-score   support

           0       0.91      0.91      0.91       483
           1       0.87      0.86      0.86       317

    accuracy                           0.89       800
   macro avg       0.89      0.89      0.89       800
weighted avg       0.89      0.89      0.89       800

0.89125
```

SGM GPU Dataset



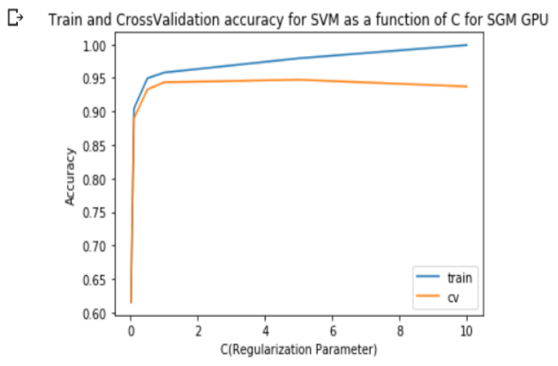
```
[59] ## Classification report for SVM-Linear
print(confusion_matrix(Y_test,y_pred_linear))
print(classification_report(Y_test,y_pred_linear))
print(accuracy_score(Y_test,y_pred_linear))
```

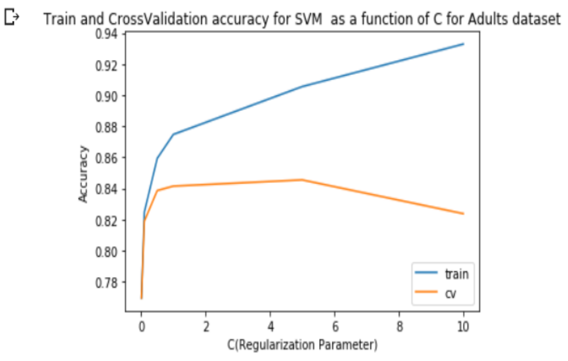
```
[[1058   73]
 [ 159  210]]
              precision    recall  f1-score   support

           0       0.87      0.94      0.90      1131
           1       0.74      0.57      0.64       369

    accuracy                           0.85      1500
   macro avg       0.81      0.75      0.77      1500
weighted avg       0.84      0.85      0.84      1500

0.8453333333333334
```

Adult Dataset

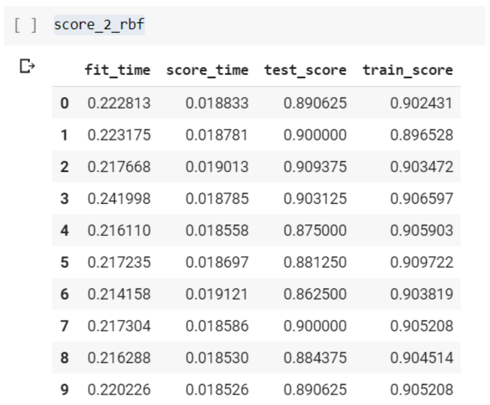
FOR rbf KERNEL:

Error rates :





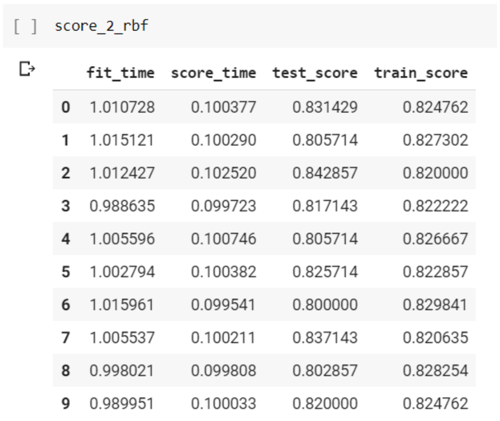
Clock Times :



```
[ ] score_2_rbf
```

	fit_time	score_time	test_score	train_score
0	0.222813	0.018833	0.890625	0.902431
1	0.223175	0.018781	0.900000	0.896528
2	0.217668	0.019013	0.909375	0.903472
3	0.241998	0.018785	0.903125	0.906597
4	0.216110	0.018558	0.875000	0.905903
5	0.217235	0.018697	0.881250	0.909722
6	0.214158	0.019121	0.862500	0.903819
7	0.217304	0.018586	0.900000	0.905208
8	0.216288	0.018530	0.884375	0.904514
9	0.220226	0.018526	0.890625	0.905208

SGM GPU Dataset

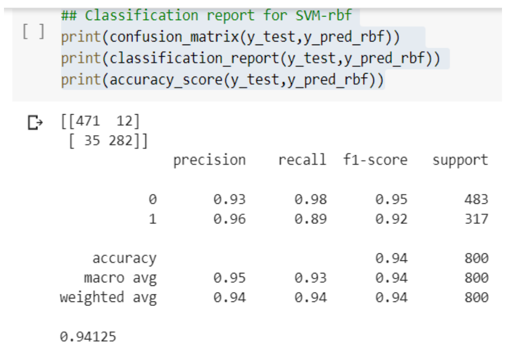


```
[ ] score_2_rbf
```

	fit_time	score_time	test_score	train_score
0	1.010728	0.100377	0.831429	0.824762
1	1.015121	0.100290	0.805714	0.827302
2	1.012427	0.102520	0.842857	0.820000
3	0.988635	0.099723	0.817143	0.822222
4	1.005596	0.100746	0.805714	0.826667
5	1.002794	0.100382	0.825714	0.822857
6	1.015961	0.099541	0.800000	0.829841
7	1.005537	0.100211	0.837143	0.820635
8	0.998021	0.099808	0.802857	0.828254
9	0.989951	0.100033	0.820000	0.824762

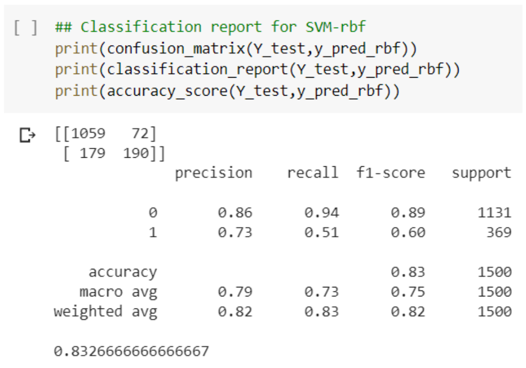
Adult Dataset

Performance Metrics :



```
[ ] ## Classification report for SVM-rbf
    print(confusion_matrix(y_test,y_pred_rbf))
    print(classification_report(y_test,y_pred_rbf))
    print(accuracy_score(y_test,y_pred_rbf))

[[471  12]
 [ 35 282]]
              precision    recall  f1-score   support

           0       0.93      0.98      0.95       483
           1       0.96      0.89      0.92       317

    accuracy                           0.94       800
   macro avg       0.95      0.93      0.94       800
weighted avg       0.94      0.94      0.94       800

0.94125
```

SGM GPU Dataset

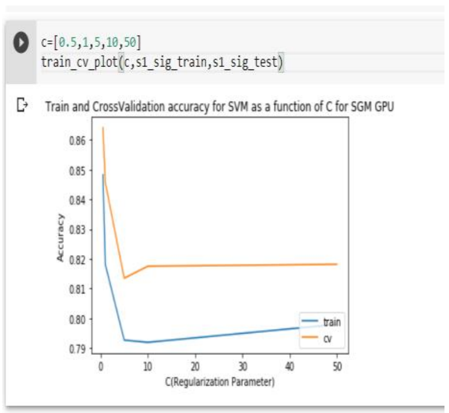


```
[ ] ## Classification report for SVM-rbf
    print(confusion_matrix(Y_test,y_pred_rbf))
    print(classification_report(Y_test,y_pred_rbf))
    print(accuracy_score(Y_test,y_pred_rbf))

[[1059   72]
 [ 179  190]]
              precision    recall  f1-score   support

           0       0.86      0.94      0.89      1131
           1       0.73      0.51      0.60       369

    accuracy                           0.83      1500
   macro avg       0.79      0.73      0.75      1500
weighted avg       0.82      0.83      0.82      1500

0.8326666666666667
```

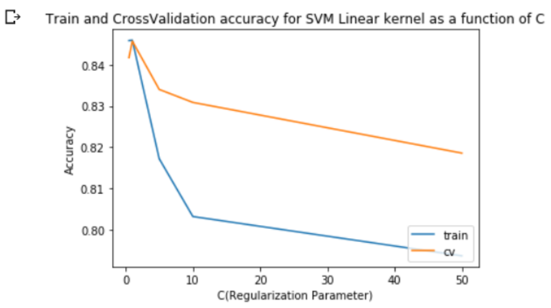
Adult Dataset

FOR sigmoid KERNEL :

Error Rates :

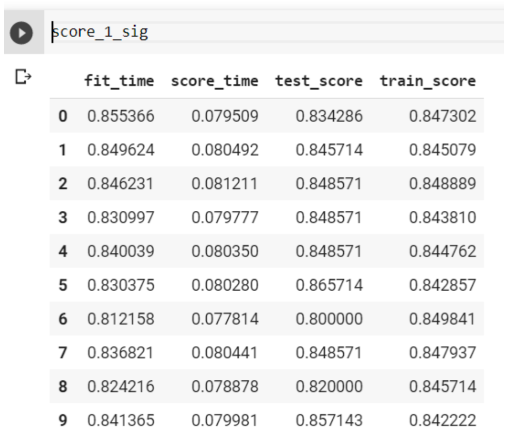


```
c=[0.5,1,5,10,50]
train_cv_plot(c,s1_sig_train,s1_sig_test)
```



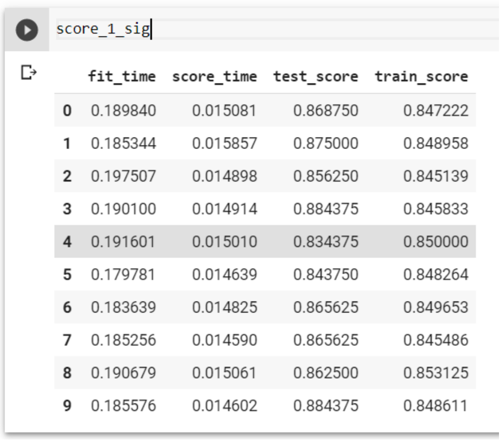
SGM GPU Dataset Adult Dataset

Clock Times :



```
score_1_sig
```

	fit_time	score_time	test_score	train_score
0	0.855366	0.079509	0.834286	0.847302
1	0.849624	0.080492	0.845714	0.845079
2	0.846231	0.081211	0.848571	0.848889
3	0.830997	0.079777	0.848571	0.843810
4	0.840039	0.080350	0.848571	0.844762
5	0.830375	0.080280	0.865714	0.842857
6	0.812158	0.077814	0.800000	0.849841
7	0.836821	0.080441	0.848571	0.847937
8	0.824216	0.078878	0.820000	0.845714
9	0.841365	0.079981	0.857143	0.842222

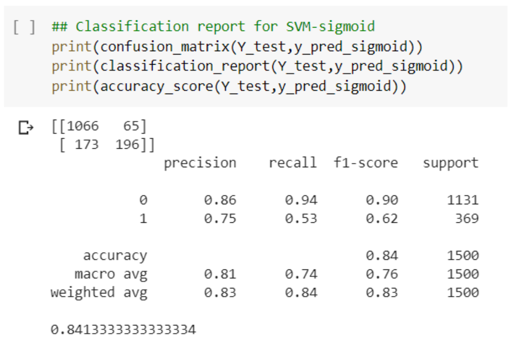


```
score_1_sig
```

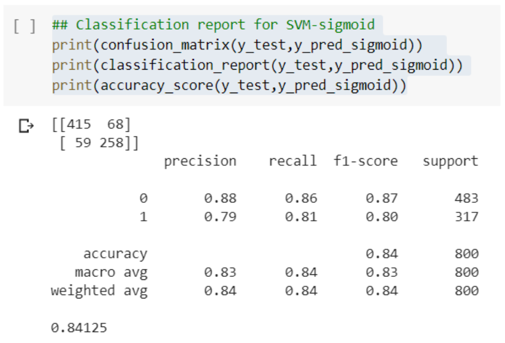
	fit_time	score_time	test_score	train_score
0	0.189840	0.015081	0.868750	0.847222
1	0.185344	0.015857	0.875000	0.848958
2	0.197507	0.014898	0.856250	0.845139
3	0.190100	0.014914	0.884375	0.845833
4	0.191601	0.015010	0.834375	0.850000
5	0.179781	0.014639	0.843750	0.848264
6	0.183639	0.014825	0.865625	0.849653
7	0.185256	0.014590	0.865625	0.845486
8	0.190679	0.015061	0.862500	0.853125
9	0.185576	0.014602	0.884375	0.848611

SGM GPU Dataset Adult Dataset

Performance Metrics :



```
[ ] ## Classification report for SVM-sigmoid
    print(confusion_matrix(Y_test,y_pred_sigmoid))
    print(classification_report(Y_test,y_pred_sigmoid))
    print(accuracy_score(Y_test,y_pred_sigmoid))

    [[1066   65]
     [ 173  196]]
                  precision    recall  f1-score   support

               0       0.86      0.94      0.90      1131
               1       0.75      0.53      0.62       369

        accuracy                           0.84      1500
       macro avg       0.81      0.74      0.76      1500
    weighted avg       0.83      0.84      0.83      1500

    0.8413333333333334
```



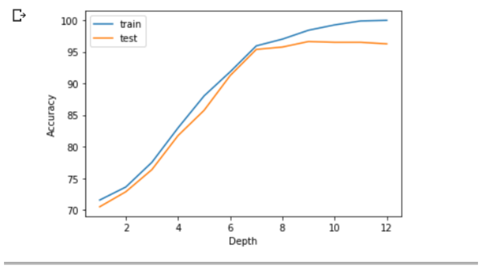
```
[ ] ## Classification report for SVM-sigmoid
    print(confusion_matrix(y_test,y_pred_sigmoid))
    print(classification_report(y_test,y_pred_sigmoid))
    print(accuracy_score(y_test,y_pred_sigmoid))

    [[415  68]
     [ 59 258]]
                  precision    recall  f1-score   support

               0       0.88      0.86      0.87       483
               1       0.79      0.81      0.80       317

        accuracy                           0.84       800
       macro avg       0.83      0.84      0.83       800
    weighted avg       0.84      0.84      0.84       800

    0.84125
```

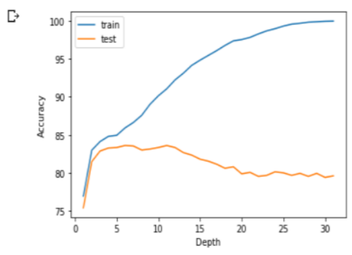
SGM GPU Dataset Adult Dataset

It can be seen that the best performance metrics were obtained overall for 'rbf' kernel.

2. **Decision Trees**

Decision tree algorithm was performed using gini index. Gini index is used here because it is computationally less intensive compared to entropy/information gain, which uses logarithmic functions. The model gave an accuracy of 80 percent on the adult dataset and an accuracy of 96.5 percent on SGM GPU dataset. Thus the model performed better on the kernel dataset. The plot of error rates for the two algorithms is as follows:

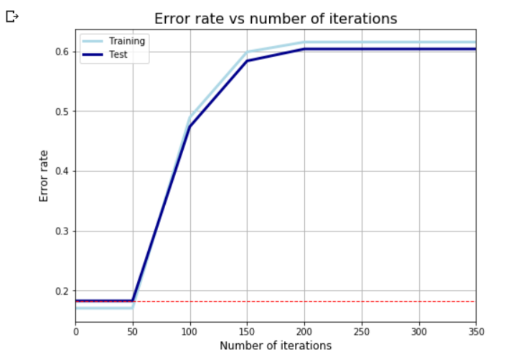


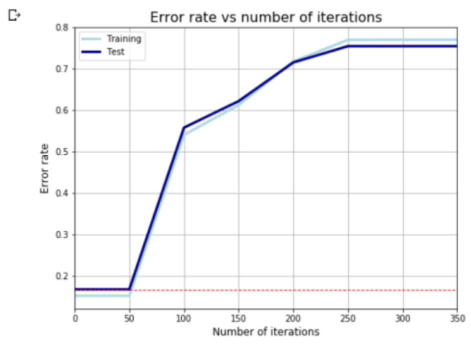


SGM GPU Dataset Adult Dataset

3. Boosting

Boosting was applied on both the datasets. AdaBoost algorithm was employed because it will consider the full weight of the classifiers and therefore predict the results with higher precision for both the datasets. A plot of Error rates as a function of both number of iterations and x was plotted which is as under.





SGM GPU Dataset Adult Dataset

As can be seen, the error rates for training and test data for both the datasets differs only marginally, meaning that the algorithm generalizes the data very well with low bias and low variance..

How does cross-validation help : Cross validation greatly reduces variance of the model, which results in improved accuracy. As can be seen in all the three algorithms, the accuracy rate obtained after applying cross-validation on the datasets results in improved accuracy.