Shalini Singh
Sxs180240

ML_Assignment2

Executive Summary

This project aims at the implementation of 3 machine learning algorithms – Support Vector Machines, Decision Trees and Gradient Boosting for 2 different datasets. A variety of control parameters are looked at for experimentation purposes for the mentioned algorithms such as *penalty, gamma,degree* for SVM; *maximum depth and pruning* for Decision Trees, *Number of boosting stages, pruning* for Gradient Boosting etc. The relationship between the variables and the accuracy/error rate is captured using appropriate plots/learning curves. The entire project is implemented using 'Python 3' (Jupyter Notebook).

Tasks Description

Task 1a – Dataset 1 – Sgemm_data

- Dataset consists of 241600 observations on 15 variables.
- The description of the variables can be found out by going to the dataset link .
- The dependent variable for the linear regression model is Avgcnt(y): Average GPU run time.
- None of the column contains any missing value, so no missing value imputation is required.

The features – 'Run 1', 'Run 2', and 'Run 3' and 'Run 4 are dropped as the average of these four parameter are calculated and used as target variable

Task 1b – Dataset 2 – Audit data

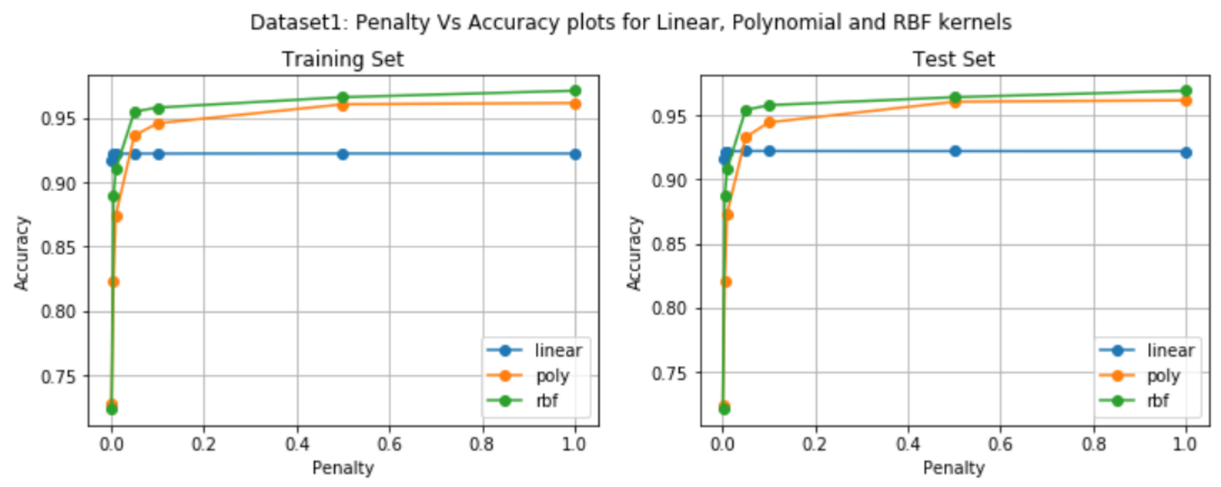
The second dataset I have chosen is an Audit dataset from Kaggle. The Audit risk dataset consists of data of various firms and their risk factors. They belong to a multitude of sectors ranging from Irrigation, Public Health, Animal Husbandry to Fisheries, Tourism, Science and Technology. This dataset is developed by a 3rd party Audit company who wishes to calculate and assess risk by analysing the present and historical risk factors thereby facilitating the audit-planning process. The dataset has over 18 columns, with one 'target' variable – 'Risk' (binary).
The primary reason I found this dataset interesting is because I have a strong interest in fraud detection. Moreover, I consider that implementing these algorithms on this dataset will expose me to understand what are the factors of importance that significantly contribute in assessing the 'Risk' of a firm.

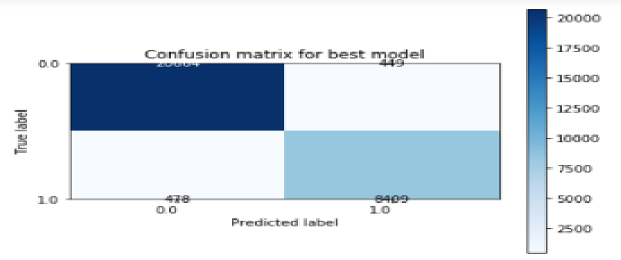
Task 2 – Support Vector Machines

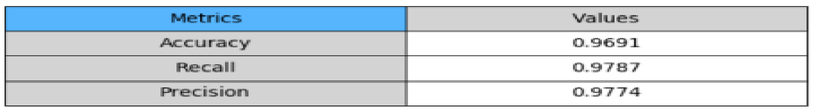
This algorithm is implemented using 'svm' package from 'scikit-learn' library. I have performed 3 experiments on SVM. I have considered linear, polynomial and rbf kernels because most non-linear datasets can be made linearly separable with one of these kernels (with proper control parameters).

Experiment 2.1 – Analysis of ' penalty ' parameter for various kernels

Penalty parameter decides the amount of allowable misclassification and consequently, the width of the margin. If penalty is high, then penalty for misclassification is high, hence margin is small and misclassification rate is small.
Dataset 1 –Rbf kernel is the best kernel as it records best performance. The best value of penalty is 1 as all models records higher train/test accuracies for that value of penalty parameter. (Plots below

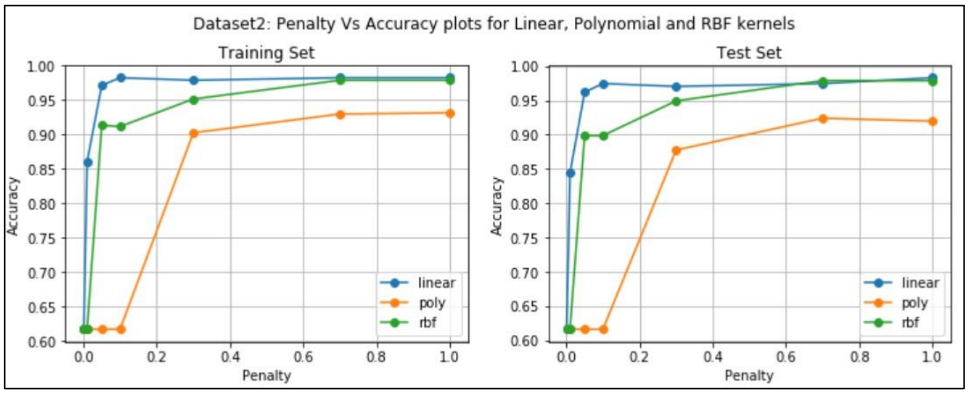


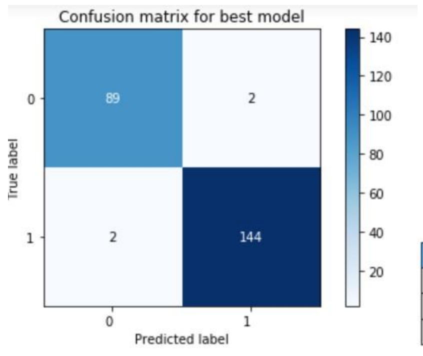


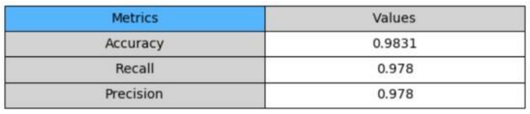


Metrics	Values
Accuracy	0.9691
Recall	0.9787
Precision	0.9774

Dataset 2 – The linear kernel is the best kernel as it records the best train/test accuracies for varying values of Penalty parameter. The best value for the control parameter is 1 as the test accuracies are the best for this value for all the models. The best model's classification metrics are reported below.





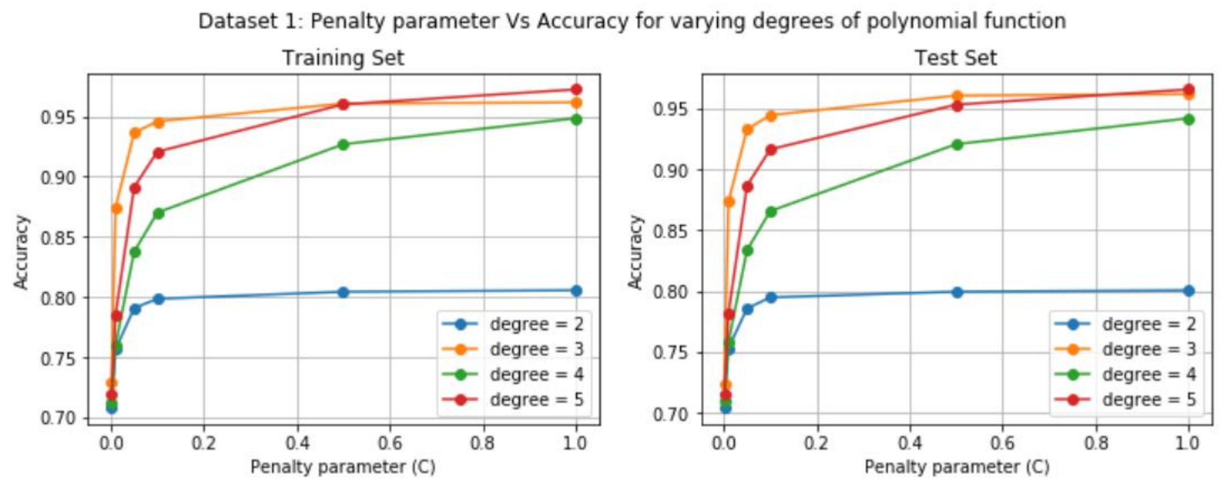


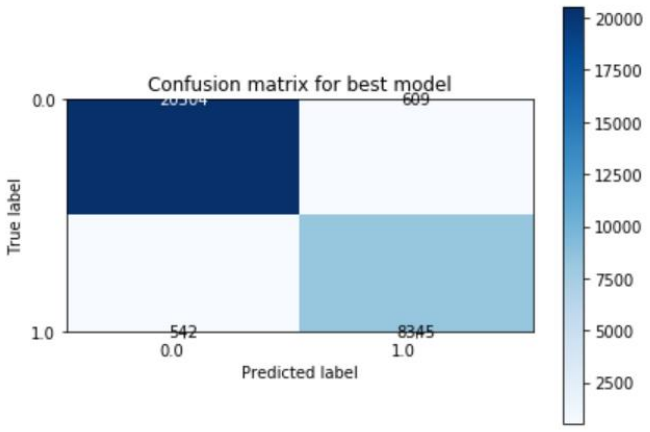
Metrics	Values
Accuracy	0.9831
Recall	0.978
Precision	0.978

Experiment 2.2 – Analysis of ' degree' parameter for Polynomial kernel

Though polynomial kernel did not record best performance in earlier experiment, I want to experiment if higher degrees of polynomial kernel can fit the data better with varying values of penalty parameter.

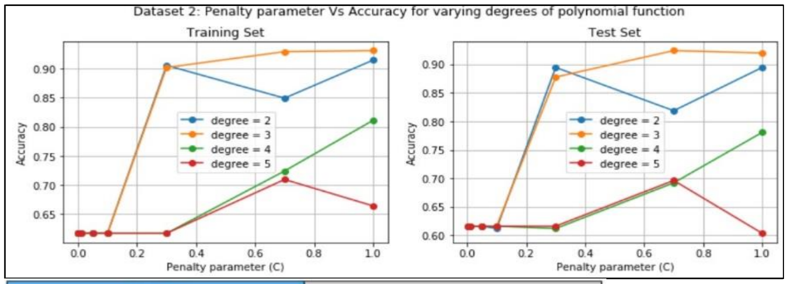
Dataset 1 – The polynomial with degree 3 and 5(higher) has the best performance with respect to Train and Test accuracies for a penalty parameter value of 1. The classification metrics are reported below.

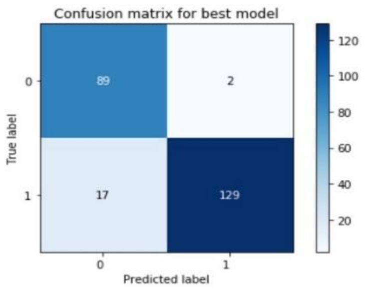




Metrics	Values
Accuracy	0.9616
Recall	0.9712
Precision	0.9742

Dataset 2 – Here again, polynmial with degree 2 outperforms other models as it records best accuracy values for penalty parameter of 1. Still, this model is not better than the 'linear' model in Experiment 2.1. The test accuracy of the previous model is 98.31% > 91.98% which is accuracy of current model. The classification metrics are reported below for reference



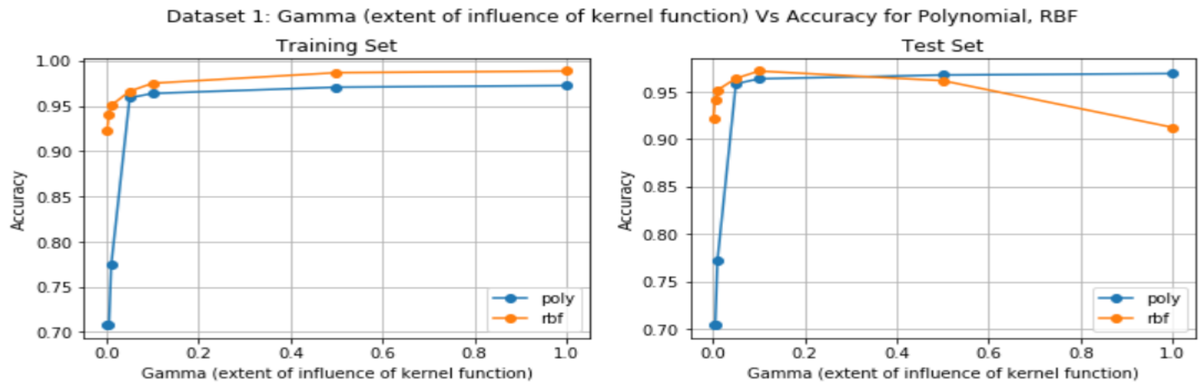


Metrics	Values
Accuracy	0.9198
Recall	0.978
Precision	0.8396

Experiment 2.3 – Analysis of ' gamma' parameter for Polynomial, RBF kernel

'Gamma' is another control parameter for SVM which signifies the 'extent of influence of the kernel function'. My intention behind this experiment is to find if there is a model that can fit the model better than the previous models by varying the 'gamma' control parameter for polynomial and rbf kernels. As 'gamma' increases the decision boundaries become more wiggled as compared to lower gamma values.

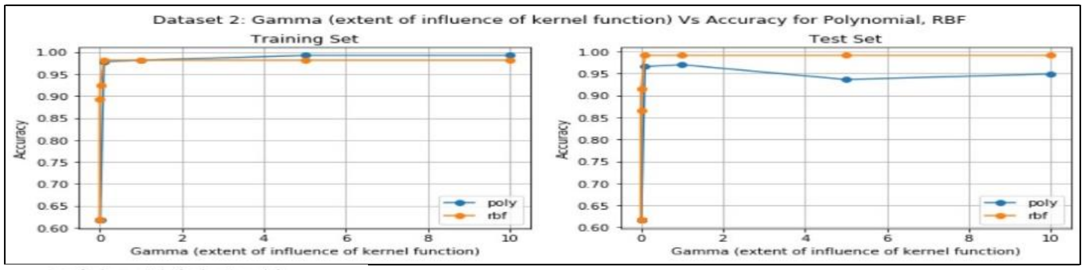
Dataset 1 – The RBF model outperforms the polynomial model slightly looking at the test accuracy metric at gamma = 1. Here, rbf accuracy decreases with the increase in value of gamma parameter for test dataset.

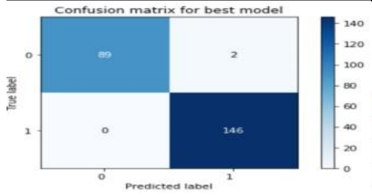


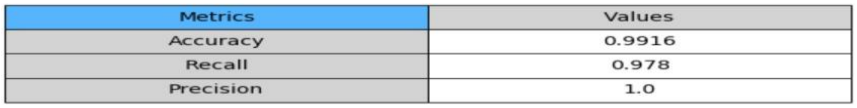
Metrics	Values
Accuracy	0.9124
Recall	0.9702
Precision	0.9111

Dataset 2 – The RBF kernel performs better than the polynomial kernel in terms of test accuracy. And, like in dataset 1, we have a better model as compared to the model obtained in Experiment 2.1.

- Current test accuracy 99.16% > 98.31% (Experiment 2.1 model)







Metrics	Values
Accuracy	0.9916
Recall	0.978
Precision	1.0

The best ‘Gamma’ value is 1 for w hic h the best accuracy metrics are recorded in above p
lots for both dataset

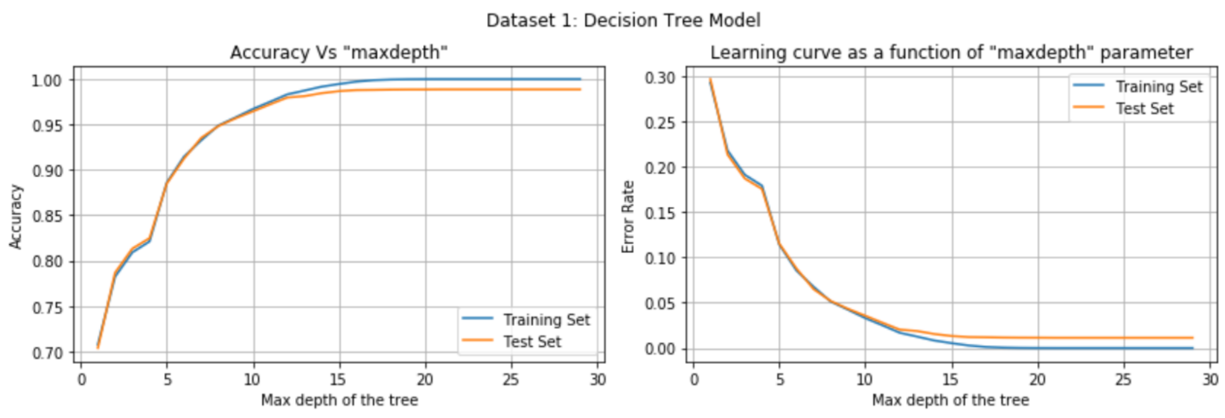
Task 3 – Decision Tree Implementation

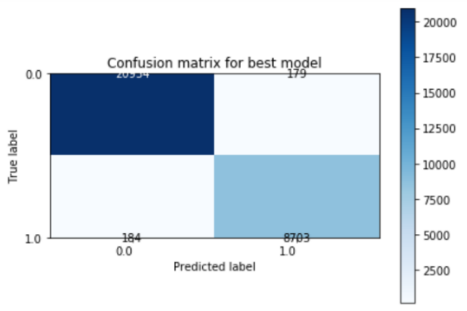
The Decision Tree model is implemented using ‘DecisionTreeClassifier’ package from ‘scikit-learn’ library. I have chosen ‘Entropy’ (Information Gain) as the splitting criterion. The primary reason for going with Entropy is it gives us a better sense of impurity at each level for binary classification problems. Gini could be more useful when the target variable is continuous nature. For this task, I have performed 2 experiments involving control parameters *maxdepth* and *pruning size* .

Experiment 3.1 – Learning Curves as a function of ‘maxdepth’

The control parameter ‘*maxdepth*’ denotes the depth of the decision tree while it is being grown. As ‘*maxdepth*’ increases, the training accuracy of the model increases, but past a point the model starts to overfit resulting in poor test accuracy. Obtaining the ideal ‘*maxdepth*’ value through learning curves is the intention behind this experiment.

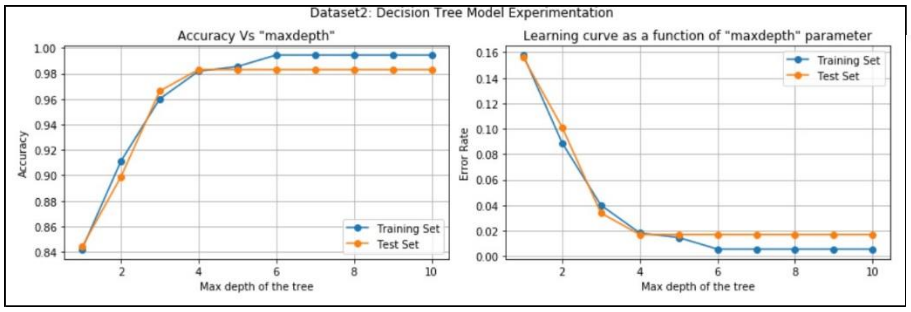
Dataset 1 - Past the value of 16, even though the training accuracy increases, the validation accuracy saturates and slightly fluctuates. Hence the ideal ‘*maxdepth*’ for Dataset1 is 16.

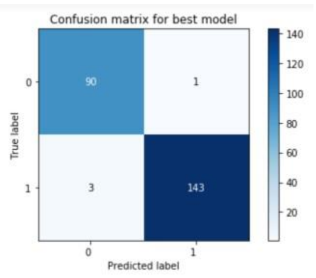




Metrics	Values
Accuracy	0.9879
Recall	0.9915
Precision	0.9913

Dataset 2 - For Dataset 2, interperting similarly, it can be seen that past the value of 4, the test accuracy saturates eventhough training accuracy keeps increasing indicating the condition of overfitting. Hence, the ideal 'maxdepth' value is 4 for this model. The classification matrix for this model is plotted above. It records a test accuracy of 98.31% with a Recall of 98.9%.



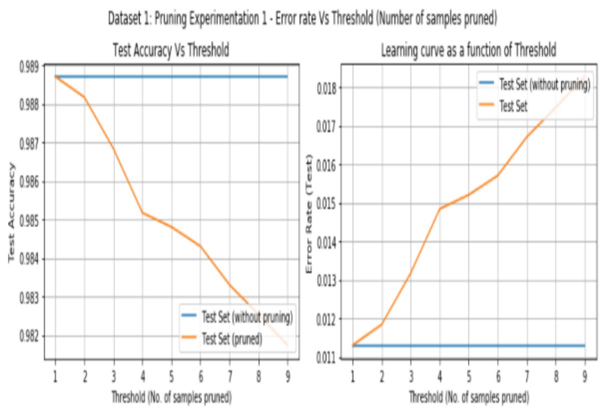


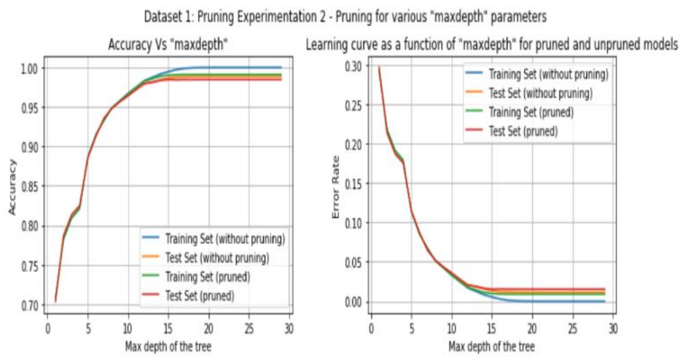
Metrics	Values
Accuracy	0.9831
Recall	0.989
Precision	0.9677

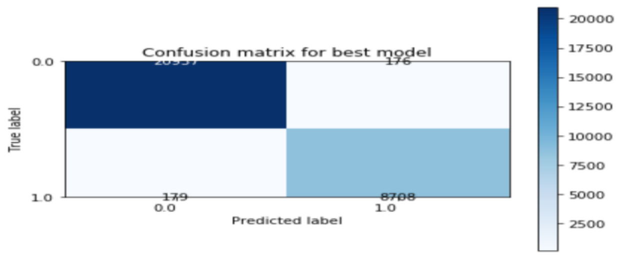
<u>Experiment 3.2 – Pruning</u>

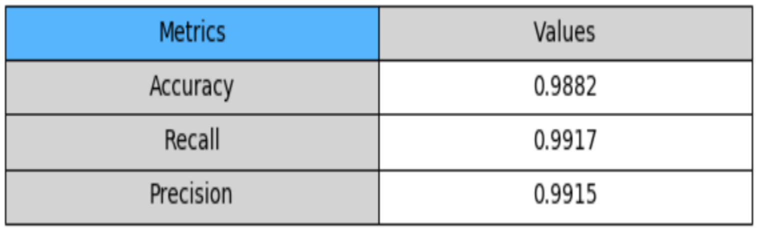
In this experiment, I have chosen 'minimum number of samples in a leaf node' as pruning condition as it is simplistic and easy to analyze. If a leaf node has the number of samples below the "*threshold*", it is cut and taken off the decision tree. Intention behind this experiment to understand effect of pruning on overfitted models.

Dataset 1 - At <u>threshold=1</u>, the test error is the least and it performs better than the original unpruned model. For this experiment, the maxdepth was not set to specify a fully grown decision tree. The experiment is performed for varying '*maxdepth'* values and fixed pruning threshold of 2. The below plot tells us that as 'overfitting' starts occuring (past 15), the pruned model performs reasonably similar to the original model.



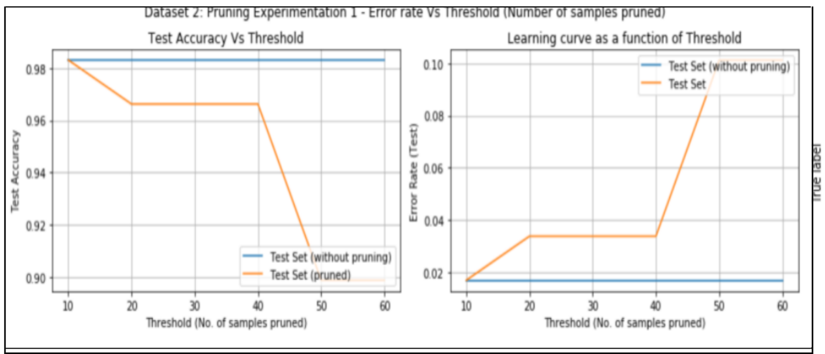


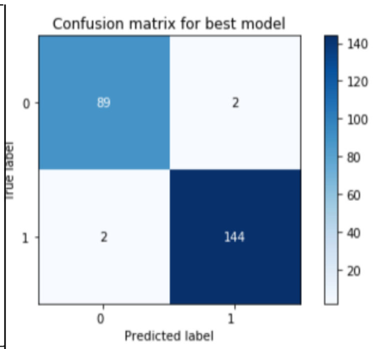


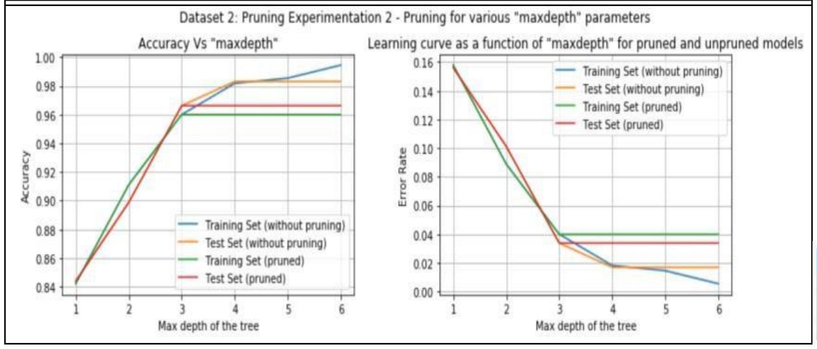


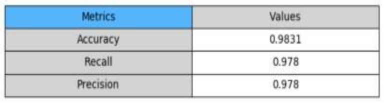
Metrics	Values
Accuracy	0.9882
Recall	0.9917
Precision	0.9915

Dataset 2 - The pruned model performs reasonably better at *'threshold'*=10. The test accuracy is significantly high at 98.31%. Also, it can be interpreted that, for a pruning with threshold of 10, the pruned test set performs reasonably similar to the test set (without pruning) until a maxdepth of 3









Metrics	Values
Accuracy	0.9831
Recall	0.978
Precision	0.978

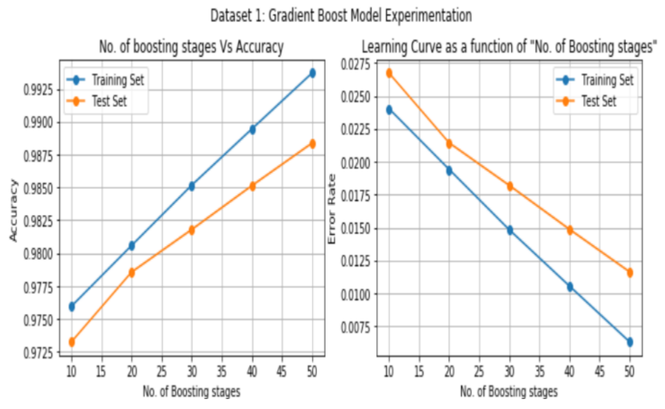
Task 4 – Boosted version of Decision Trees

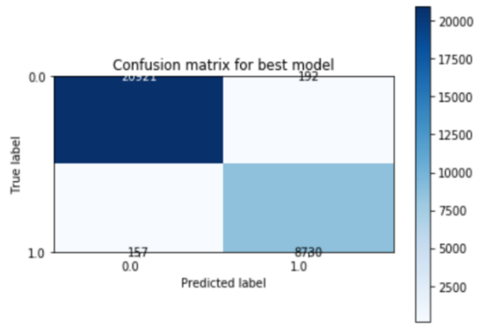
In this task, I have implemented the Gradient boost model for Decision Trees using the 'GradientBoostingClassifier' package from 'scikit-learn' library. I have performed 2 experiments using control parameters – <u>Number of boosting stages</u> and <u>pruning</u>.

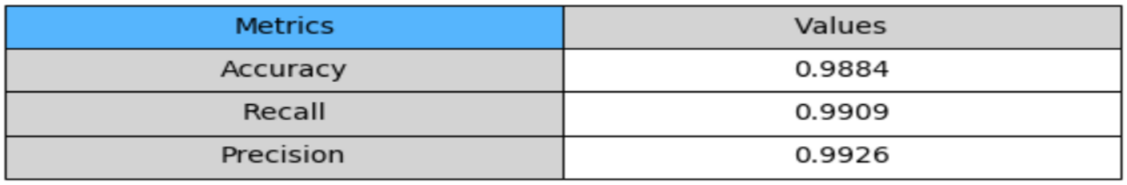
<u>Experiment 4.1 – Experimenting with 'Number of boosting stages'</u>

'Boosting stages' refers to the number of times the model is sequentially worked upon on its misclassification. The intention behind this experiment is to understand the relationship between the control parameter and train/test error rates.

Dataset 1 - In the considered range, <u>50 is the best value</u> for the control parameter as evident by the highest test/train accuracies. The classification metrics for this model is reported below

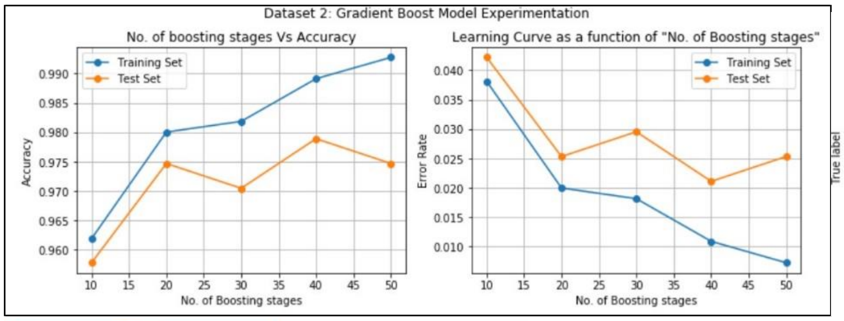


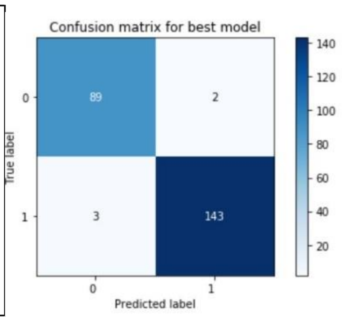


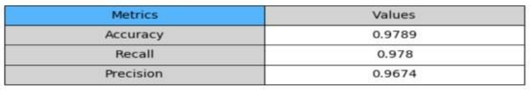


Metrics	Values
Accuracy	0.9884
Recall	0.9909
Precision	0.9926

Dataset 2 - As ‘No. of boosting stages’ increases, the test accuracy fluctuates exhibiting no pattern. The best value for the <u>control parameter is 40</u>, for which the test accuracy is 97.89%.





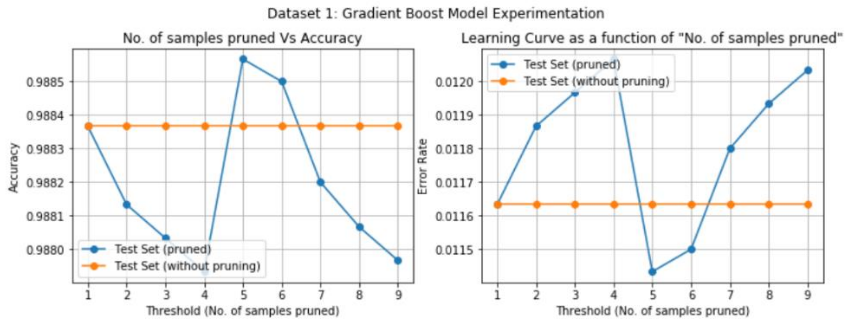


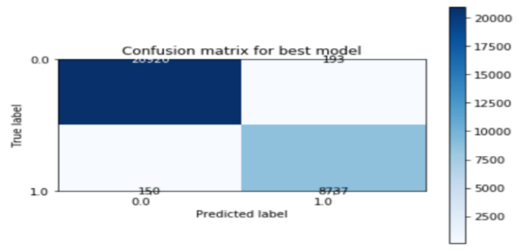
Metrics	Values
Accuracy	0.9789
Recall	0.978
Precision	0.9674

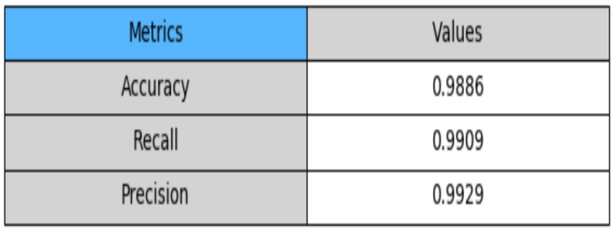
Experiment 4.2 – Experimenting with Pruning

Pruning is performed by cutting nodes with less than ‘threshold’ number of samples off the tree.

Dataset 1 – <u>Best model is when threshold = 5</u>, recording test accuracy of 89.02% and performing better than the original unpruned model.

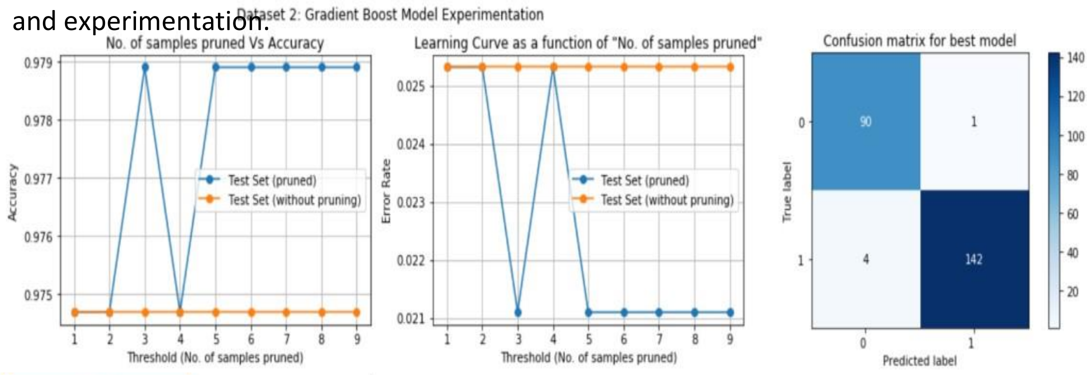


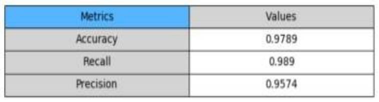




Metrics	Values
Accuracy	0.9886
Recall	0.9909
Precision	0.9929

Dataset 2 - The best model is when the control parameter is 5. Key takeaway from this experiment is that pruning does help in arriving at a better model. The best model is arrived at through learning curves and experimentation.

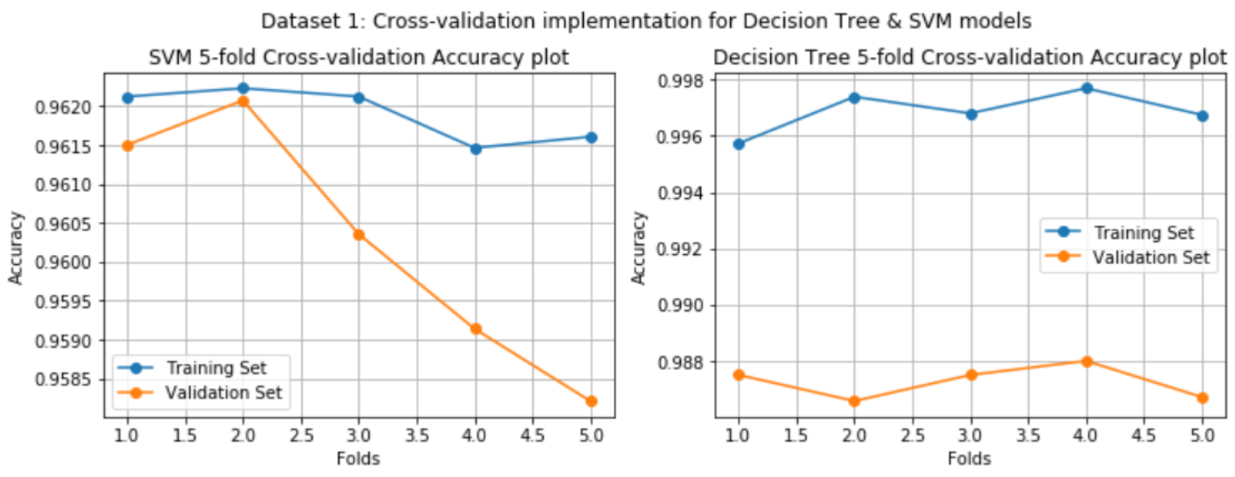




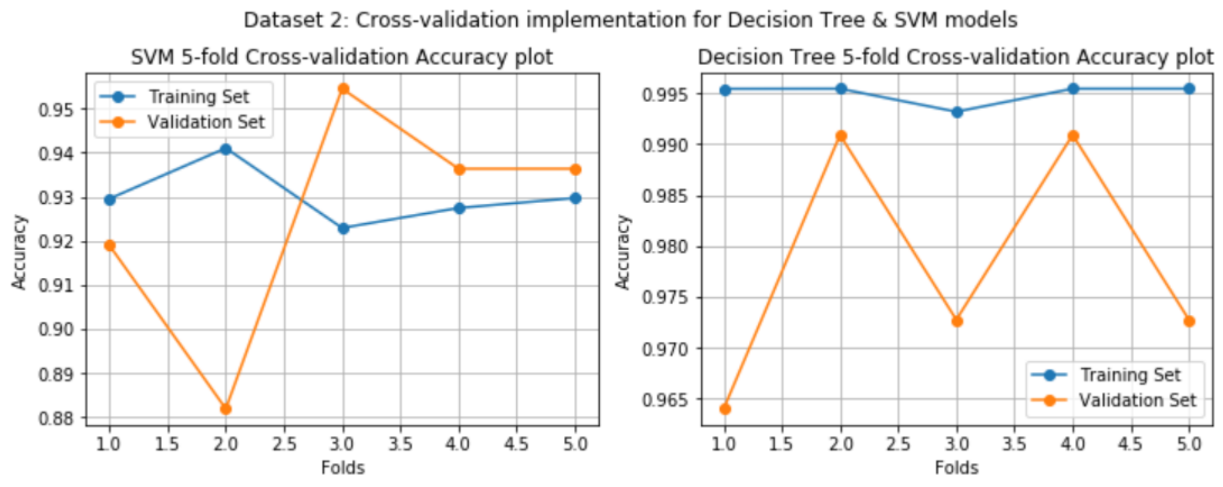
Metrics	Values
Accuracy	0.9789
Recall	0.989
Precision	0.9574

Task 5 – Cross Validation Implementation

5-fold cross-validation is implemented for both datasets . This experiment is performed to understand the importance of cross-validation sets. For dataset 1, the 5th fold records very low train/test error at **folds = 2 for SVM and folds = 4 for decision Tree** and produces the best model for both SVM and Decision Tree models.



For dataset 2, for SVM, the 5th fold produces the best model because the train and test accuracies are comparable and don't exhibit extremities. For Decision Tree, 2nd and 4th fold produce the best results for dataset 2.



Conclusion / Discussion

- <u>Dataset 1 – Best model – Boosted model (boosting stages = 50)</u>
  The boosted model is the best model. In my opinion, this is because the dataset is dense, not linearly separable and no single model can precisely capture all the variance exhibited by the dataset. Hence, an ensembled model (a series of models) that sequentially works on misclassification of previous models is the best model.
  SVM and Decision Tree models performed reasonably similar on Dataset 1. Th SVM RBF kernel was the only model that neared the best model with an accuracy of 96.91% for a gamma value of 1. This is just an indication of how difficult to separate the data linearly.
- <u>Dataset 2 – Best model – RBF SVM (gamma = 1) – 99.16% test accuracy</u>
  The dataset has over 1k rows. Being sparse, the data is linearly separable on a higher dimension and hence, an RBF kernel is able to classify the data almost perfectly.
  SVM performed way better than Decision Tree in the 2nd dataset. This is primarily because the dataset is linearly separable. Though an ensembled model sequentially works on misclassification errors, it could not stand out as compared to SVM, the main reason being the dataset's distribution and size.
- There is no "best" algorithm on a global basis. It solely depends on the dataset, its size, distribution, variance etc. From my perspective, the "best" algorithm would have optimal training accuracy and the best testing accuracy.