**Applied Machine Learning – Assignment 2**

**INTRODUCTION**

Classification is a technique to categorize the data into a desired and distinct number of classes where we can assign label to each class. We find applications of classification in almost every industry such as speech recognition, biometric identification or in various important decisions of our life such as whether to set up a business in a location or not, to apply to a particular college or not, to buy a house in a neighborhood or not. The Classifiers can be binary classifier (with 2 distinct classes) or multi-class classifier (with more than 2 distinct classes). Several classification algorithms include logistic regression, Naïve Bayes, Support Vector Machines, K- nearest neighbor, Decision tree, and other bagging and boosting algorithms such as Random forest, AdaBoost etc.

For this assignment I have experimented with Support Vector machine, Decision trees and Boosting on two datasets.

* **Dataset1 –** The first dataset used in this assignment is the Facebook comment prediction dataset. As suggested, I have converted it to a binary classification problem by thresholding the output to a class label.

New\_Target – This is the class label that is initialized as follows:

If Target Variable > 0, New\_Target = 1 (If a post gets a comment it is labeled as 1)

If Target Variable <= 0, New\_Target = 0 (If a post does not get a comment it is labeled as 0)

This dataset is interesting as it can help people and businesses understand what features are important for increasing the popularity of a product or a brand by measuring the response to advertisement posted on Facebook in terms of comments received from the people.

* **Dataset2 –** The second dataset used is the Admission Prediction dataset. Education is one of the biggest investments of a person’s life, so every student should do proper research on various universities based on his/her skills and academic standing before applying for a particular university. The application process is tedious and requires lots of hard work, time and money, so proper research can help them save lots of time and money. In the dataset there are several qualitative and quantitative variables:

Quantitative **-** GRE Score, TOEFL Score, University Rating, SOP Score, LOR Score, CGPA, Work Experience, Age, Number of Certifications/Awards, Number of Languages, Chance of admit

Qualitative – Research paper published or not, Volunteer/Leadership experience or not, Nation/International level sports player or not, Member of an NGO or not.

Admit – This is the class label that is initialized as follows:

If Chance of Admit > 0.6, Admit = 1 (If the chance of admit is greater than 0.6, it is labeled as 1)

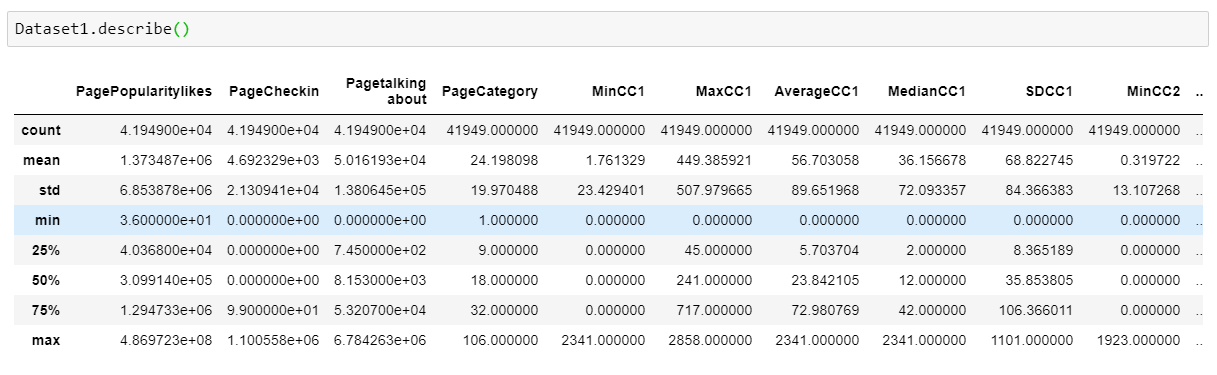
If Chance of Admit <= 0 .6, Admit = 0 (If the chance of admit is less than 0.6, it is labeled as 0)

This dataset is interesting as it can help students to take the most crucial decision of their life and will prevent them from unnecessary hassle and expenditure during the application process as they can take more sound decisions.

Experimented with different threshold values using ROC curves.

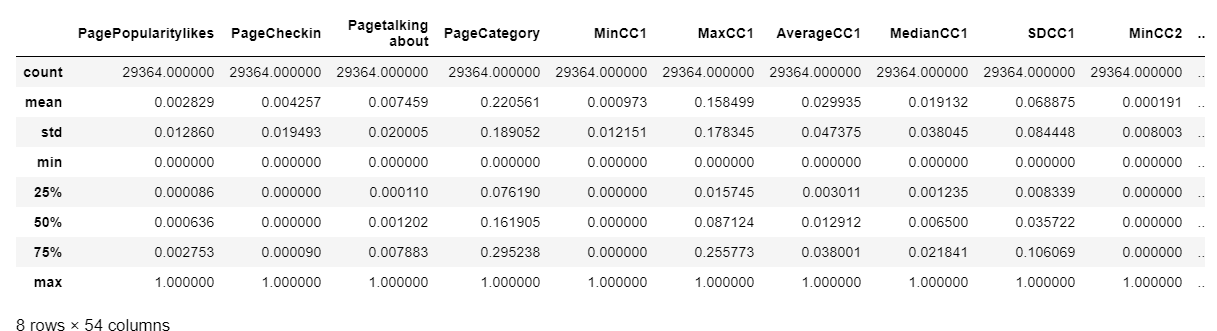
**Dataset 1(Facebook Comment Prediction Dataset)**

Performed exploratory data analysis to find the relationship between different features and to understand the distribution of features.



**Fig. 1**

We can see from Fig. 1 that range of values for different features differ a lot, so I performed feature scaling on these features.



**Fig. 2**

Fig. 2 shows the details of the scaled features.

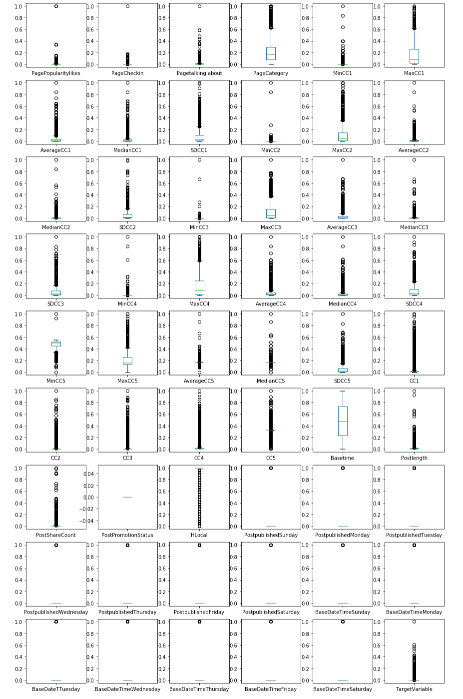
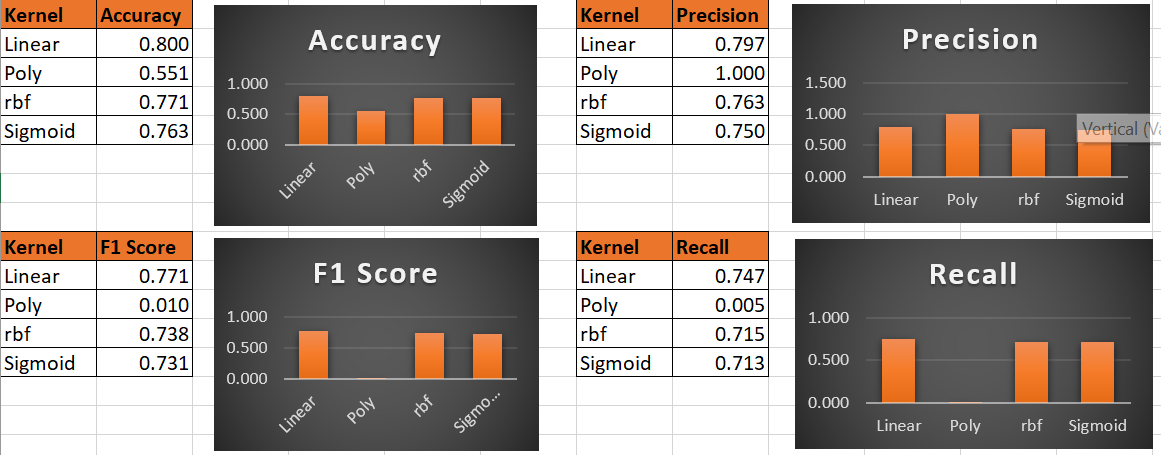


Fig. 3 shows the box plots for all the features for dataset1. We can see that there are outliers in this dataset. So, we have handled these outliers.

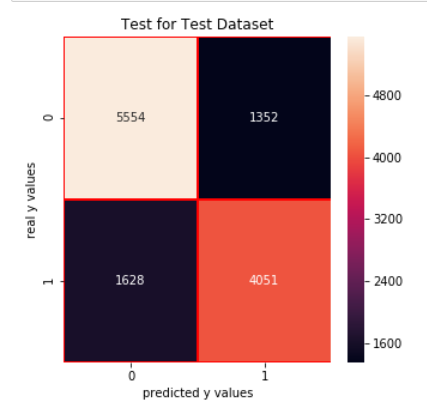
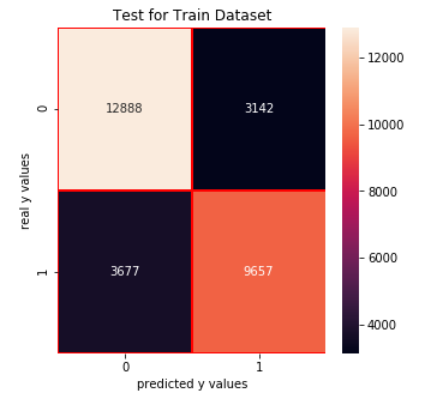
**Fig. 3**

**SVM Implementation –** Used support vector machine for classification using different kernel functions. Fig.4 shows the performance of different kernel functions on the test data. We can see that on an average linear kernel performs better than polynomial, rbf and sigmoid kernels for this dataset.



**Fig.4**

For the linear kernel accuracy score is 0.80, which means the model is approximately 80% accurate. The precision score is 0.797, recall is 0.747 and F1 score is 0.771. The results are good as per the several performance metrics for linear kernel. We can also experiment with other kernel functions and compare the model performance.

**Fig. 5**

Fig. 5 shows the confusion matrix for linear kernel for both test and train dataset.

Experimented by using **K fold cross validation with 5 folds**, for the linear kernel SVM performed like the train/test split method with the cross\_val\_score value equal to 0.795. I have decided to use Train/Test Split method for this dataset because the amount of time required to train dataset is lower than cross validation and there is no significant improvement in the model performance if I use K- fold cross validation for this dataset. Generally cross validation method performs better on the unseen data as compared to the Train/Test split.

**Decision Tree Implementation** – I have experimented with two methods “entropy” and “Gini index” to split on variables. Generally, it does not matter which method we choose they give same results most of the times. Gini Index is used to minimize misclassification and Entropy is used for exploratory analysis. Entropy takes slightly more time than Gini Index because of the log calculation.

|  |  |
| --- | --- |
| **Method** | **Accuracy Score** |
| Entropy | 1.000 |
| Gini Index | 0.978 |

**Fig. 6**

From Fig. 6 we can see that the accuracy for Entropy method is more than that of Gini Index. And since for our dataset the time taken was approximately the same by both the methods, I have chosen Entropy for future experiments on this dataset.

**Fig. 7 Fig. 8**

Fig. 7 and Fig. 8 shows the effect of the different tree depths on the accuracy of the model. We can see as we increase the depth of the decision tree the accuracy increases, but this might lead to overfitting in several scenarios. So, we usually decide an appropriate depth where test and train error are minimum or have good performance measures. We don’t face this issue in case of this dataset, the non – pruned decision tree model performs equally good on the test data as it does in the case of training data. We normally continue to prune the tree if it reduces the error significantly. So, we can take the depth of the tree as 12, the accuracy at that point is 0.97 after that the accuracy is fluctuating a little. I have considered only the maximum depth of the tree for pruning for this dataset, we can also control for maximum number of leaf nodes, minimum number of samples required on the leaf node and some other parameters as well to improve the performance of the model.

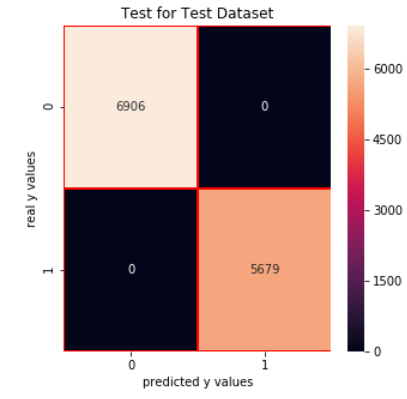


Fig. 9 shows the confusion matrix for the decision tree without pruning. We can see that the values of accuracy score, precision, recall and f1 score are all 1 for this dataset.

**Fig. 9**

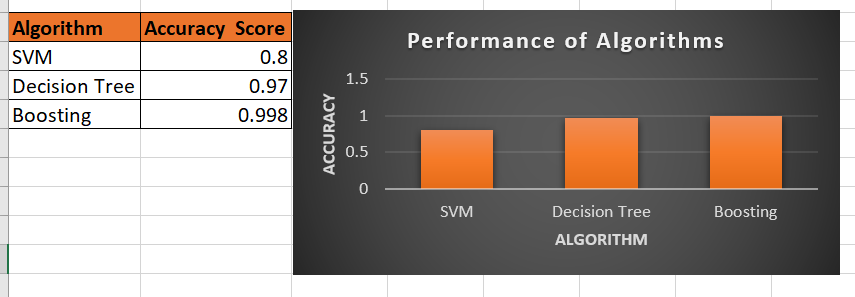
**Boosting Implementation –** I have used AdaBoost as the boosted version of the decision tree. The accuracy score without pruning is 1. We can take the depth of the tree as 5, the accuracy of that point is 0.9988 after that it is fluctuating a bit. I have considered only the maximum depth of the tree for pruning for this dataset, we can also control for maximum number of leaf nodes, minimum number of samples required on the leaf node and some other parameters as well to improve the performance of the model.

Fig. 10 shows the accuracy score for different tree depth for the boosted version of the decision tree.

**Fig. 10**

**Comparison of algorithms –**

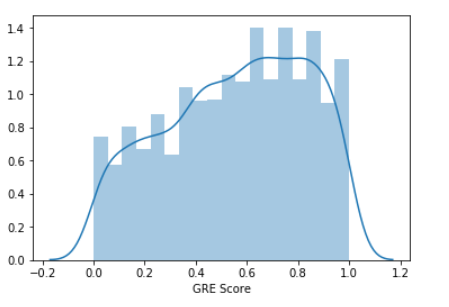
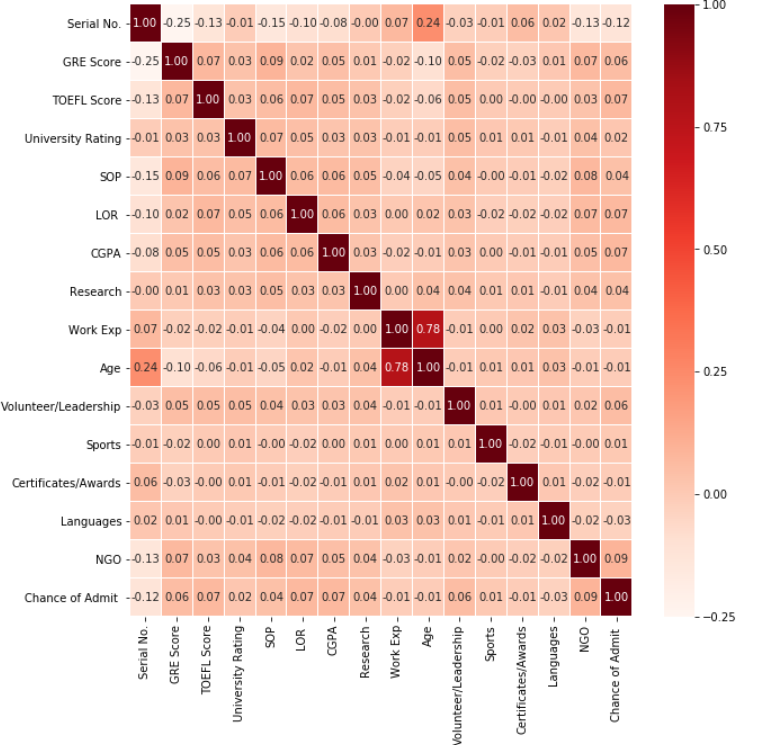
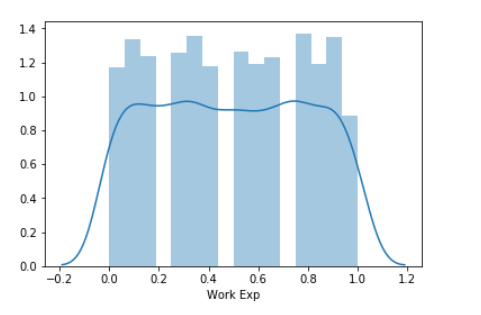
Fig. 11 shows the comparison between different algorithms that we have experimented for this dataset. We can see boosted version of the decision tree performs the best in terms of accuracy as well as other performance metrics (precision, recall, f1 score).



**Fig. 11**

**Dataset 2 (Admission Prediction dataset)**

Performed exploratory data analysis to find correlation among features:

   
 **Fig. 12**

The above graphs show that the features are scaled and now ranges from 0 to 1. More analysis on features in present in the code.

Scaled the features to bring them in the same range.

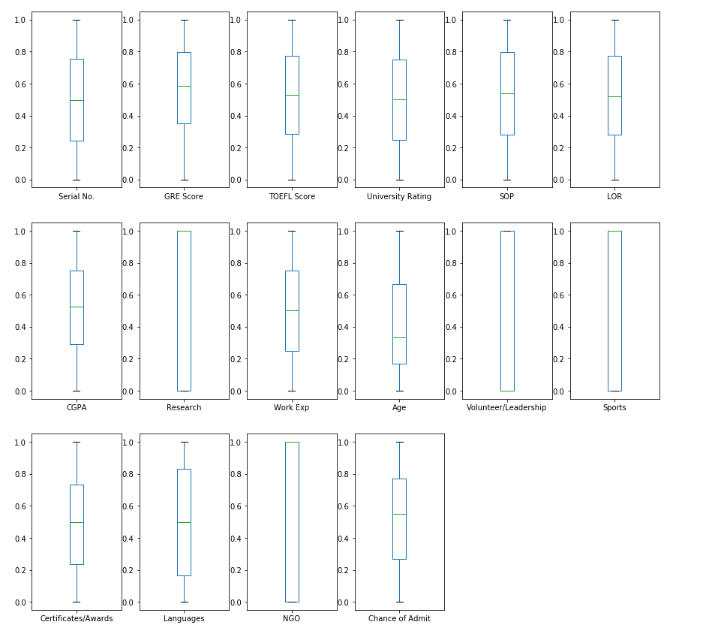
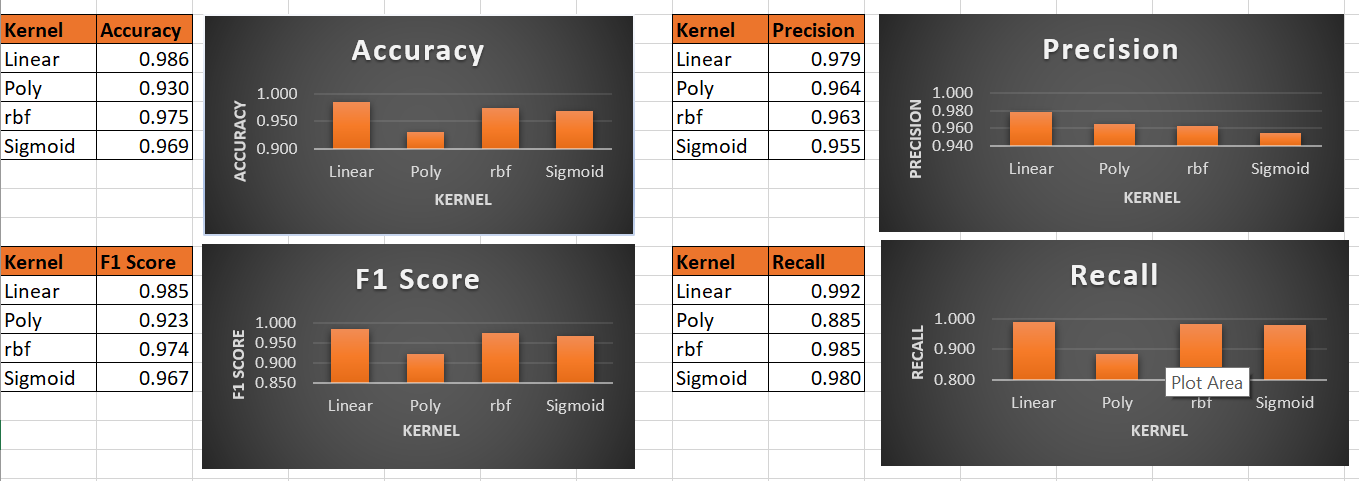


Fig. 13 shows the box plots for all the features, we can see that there are no outliers present in this dataset.

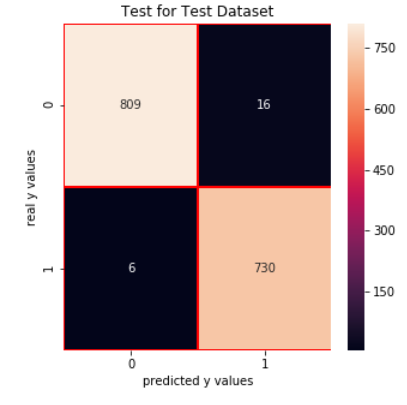
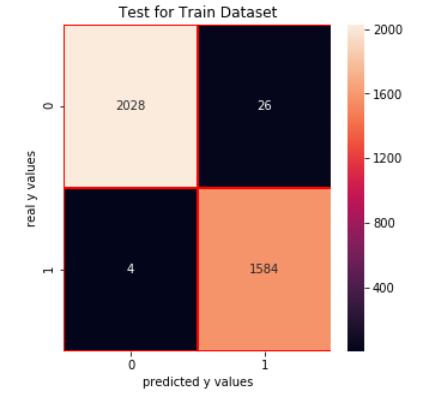
**Fig. 13**

**SVM Implementation-** Used support vector machine to classify if the student will get an admit from a university or not. Fig. 14 shows the performance metrics for different kernel function on the test data. We can see that on an average linear kernel performs better than polynomial, rbf and sigmoid kernels for this dataset.



**Fig. 14**

For the linear kernel accuracy score is 0.986, which means the model is approximately 98.6% accurate. The precision score is 0.979, recall is 0.992, F1 score is 0.985. The results are good as per the several performance metrics for linear kernels. We can also experiment with other kernel functions and compare the model performance.

**Fig. 15**

Fig. 15 shows the confusion matrix for linear kernel for both test and train dataset.

Experimented by using **K fold cross validation with 5 folds**, for the linear kernel SVM performed like the train/test split method with the cross\_val\_score value equal to 0.986. I have decided to use Train/Test Split method for this dataset because the amount of time required to train dataset is lower than cross validation and there is no significant improvement in the model performance if I use K- fold cross validation for this dataset. Generally, cross validation performs much better on the unseen data as compared to Train/Test split method. We can also experiment with different number of folds (k values) and compare the model performance.

**Decision Tree Implementation** – I have experimented with two methods “entropy” and “Gini index” to split on variables.

Fig. 16 shows that the accuracy score for both Entropy and Gini index method is 1. As entropy takes more time, I will use Gini index for rest of the experiments for this dataset.

|  |  |
| --- | --- |
| **Method** | **Accuracy Score** |
| Entropy | 1.0 |
| Gini Index | 1.0 |

**Fig. 16**

**Fig. 17 Fig. 18**

For Fig. 17 and Fig. 18 shows the effect of the different tree depths on the accuracy of the model. We can see as we increase the depth the accuracy increases, but this might lead to overfitting. So, we usually decide an appropriate depth where test and train error are minimum, or the accuracy is high. We don’t face this issue in case of this dataset, the non – pruned decision tree model performs equally good on the test data as it does in the case of training data. We normally continue to prune the tree if it reduces the error significantly. So, in this case the depth of the tree can be taken as 6 because the accuracy is 1 at that point. I have considered only the maximum depth of the tree for pruning for this dataset, we can also control for maximum number of leaf nodes, minimum number of samples required on the leaf node and some other parameters as well to improve the performance of the model.

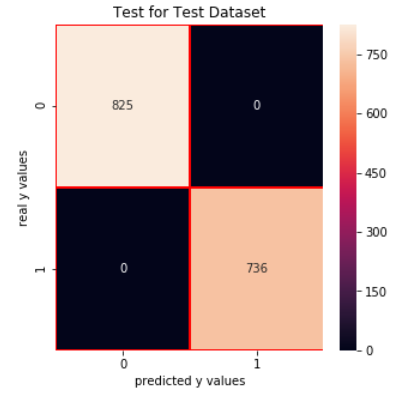


Fig. 19 shows the confusion matrix for the decision tree without pruning for dataset 2. We can see that the values of accuracy score, precision, recall and f1 score are all 1 for this dataset.

**Fig. 19**

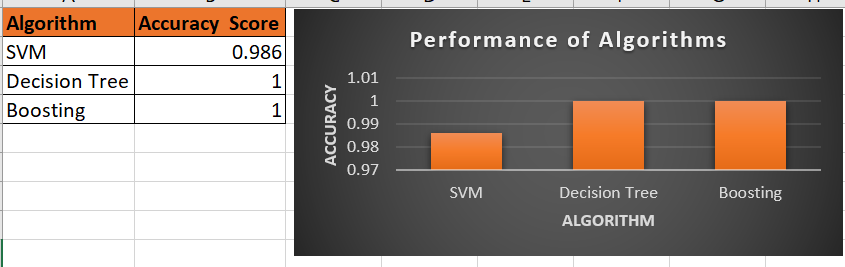
**Boosting Implementation -** I have used AdaBoost as the boosted version of the decision tree. The accuracy score without pruning is 1. We can take the tree of depth 2 as the accuracy at that point is 1.

Fig. 20 shows the accuracy score for the boosted algorithm across different tree depths.

**Fig. 20**

**Comparison of algorithms –**

Fig. 21 shows the comparison between different algorithms that we have experimented with in this assignment. We can see that both decision tree and boosted algorithm performs better than SVM in terms of accuracy for this dataset. We can use other performance metrics as well depending on the requirement to compare the model performance.



**Fig. 21**

**CONCLUSION**

Based on the experiments we can conclude that on average boosted algorithms perform better in comparison to SVM and decision trees for the two datasets. There is no one algorithm for all the types of data we should experiment with different types and different variations (by tuning the parameter values) of algorithm to find the best one for our dataset.