Prasanna Gopala Doddaballapura Project 2 6341.501

Classification Algorithms

Executive Summary

In Part-1 I have applied various classification algorithms on the Energy consumption data set and classified the houses as “high energy” consumers or “low energy” consumers.

I have dropped the lights and the date column from the dataset as they are not useful for us when doing the classification. I have split the dataset into training set and testing set with a ratio of 30% testing set and 70% training set. And standard student scaling is done on all the feature variables. Student scaled variable has a mean of 0 and standard distribution of 1.

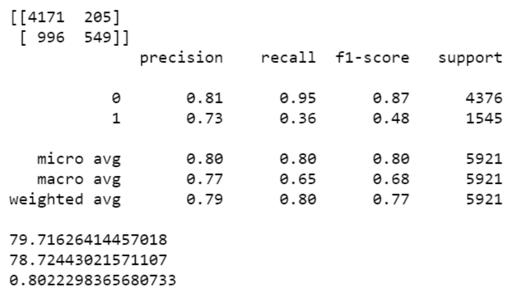
- Support Vector Classifier
  3 types of SVM’s are fitted on the dataset and the accuracy rate of each of them are provided in the table below. The confusion matrix of each of them is also provided.

- Decision Trees
  Decision tree with both “Gini index” and “information gain” as splitting criteria are fitted, and their accuracy and confusion matrices are reported.
  For experimenting with the pruning, I have performed a grid search on the various values of min_sample_leaf and max_depth and I have reported the best combination of all the parameters keeping accuracy as a scoring criterion.

  The best tree reported has an accuracy around 86% and has the parameter values set as {'criterion': 'gini', 'max_depth': 50, 'min_samples_leaf': 5}

- Boosting
  I have tested out 2 boosting algorithms namely Gradient Boosting Classifier and Ada Boost Classifier and I have reported their accuracy scores.
  As Ada Boost Classifier performs better on this dataset, I have performed grid search on the n_estimators parameter values and I have reported the best possible value for this parameter keeping accuracy as a scoring criterion.

1. Support Vector Classifier
   - Linear SVC: When we fit a Linear SVC we get the following result.

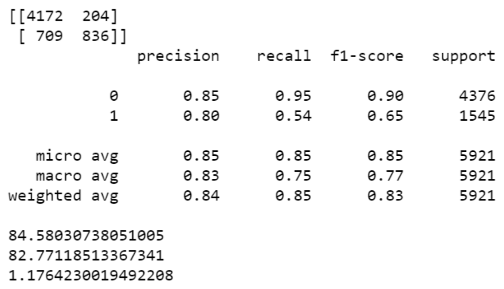


```
[[4171  205]
 [ 996  549]]
              precision    recall  f1-score   support

           0       0.81      0.95      0.87      4376
           1       0.73      0.36      0.48      1545

   micro avg       0.80      0.80      0.80      5921
   macro avg       0.77      0.65      0.68      5921
weighted avg       0.79      0.80      0.77      5921

79.71626414457018
78.72443021571107
0.8022298365680733
```

From the confusion matrix above we see that there are lot of misclassifications and from the accuracy score based on a single Test dataset is 79.71%.
I performed a cross validation with 10 folds and the mean accuracy of this 10 folds is reported to be around 78% with a standard distribution of 0.8% which is acceptable.
we also will try a non-linear classifier and check the results.

We have tried Two kernel SVC’s namely the RBF and the polynomial.

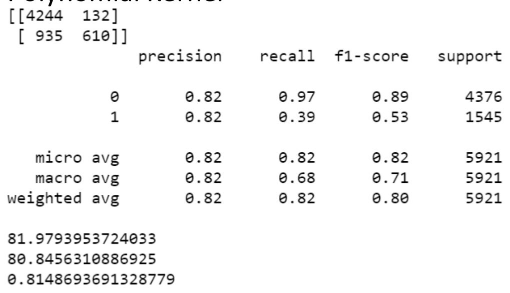
- RBF kernel



```
[[4172  204]
 [ 709  836]]
              precision    recall  f1-score   support

           0       0.85      0.95      0.90      4376
           1       0.80      0.54      0.65      1545

   micro avg       0.85      0.85      0.85      5921
   macro avg       0.83      0.75      0.77      5921
weighted avg       0.84      0.85      0.83      5921

84.58030738051005
82.77118513367341
1.1764230019492208
```

From the RBF kernel result we see that the accuracy has improved and there is less misclassification. But the specificity of the model has gone up. I performed a cross validation with 10 folds and the mean accuracy of these 10 folds is reported to be around 82.77% with a standard distribution of 1.17% which is acceptable.

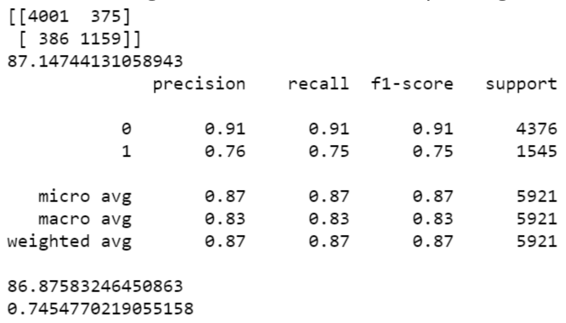
- Polynomial Kernel



```
[[4244  132]
 [ 935  610]]
              precision    recall  f1-score   support

           0       0.82      0.97      0.89      4376
           1       0.82      0.39      0.53      1545

   micro avg       0.82      0.82      0.82      5921
   macro avg       0.82      0.68      0.71      5921
weighted avg       0.82      0.82      0.80      5921

81.9793953724033
80.8456310886925
0.8148693691328779
```

From the polynomial kernel result we see that accuracy is a bit low compared to the RBF kernel but the specificity of the model which is the recall of the model has come down. I performed a cross validation with 10 folds and the mean accuracy of these 10 folds is reported to be around 81.97% with a standard distribution of 1.17% which is acceptable.

2. Decision Trees
   - Information gain as the criteria for splitting the nodes



```
[[4001  375]
 [ 386 1159]]
87.14744131058943
              precision    recall  f1-score   support

           0       0.91      0.91      0.91      4376
           1       0.76      0.75      0.75      1545

   micro avg       0.87      0.87      0.87      5921
   macro avg       0.83      0.83      0.83      5921
weighted avg       0.87      0.87      0.87      5921

86.87583246450863
0.7454770219055158
```

When we use information gain which is also called “entropy”as the splitting criteria we get the mean accuracy as around 86.8% and a standard deviation of 0.7% with 10 folds cross validation
which is much better then a Kernel RBF SVM which was the best performing SVM

- Gini Index as a criterion for splitting the nodes

```
[[4002  374]
 [ 372 1173]]
87.40077689579464
              precision    recall  f1-score   support

           0       0.91      0.91      0.91      4376
           1       0.76      0.76      0.76      1545

   micro avg       0.87      0.87      0.87      5921
   macro avg       0.84      0.84      0.84      5921
weighted avg       0.87      0.87      0.87      5921

86.97006673992085
0.9151080901673867
```

When we use the Gini index as criterion for splitting the nodes, we get the mean accuracy as around 86.97% and a standard deviation of 0.9% with 10 folds cross validation
which is just a bit lower than a using entropy decision tree. Therefore, we can conclude that there is not much difference when we use either entropy or Gini Index.

- Experimenting with pruning the decision tree
As Grid Search technique provides a easy way to know which is the best combination of the parameters., when I experimented with pruning. The various values that I have passed for the parameters are as follows.

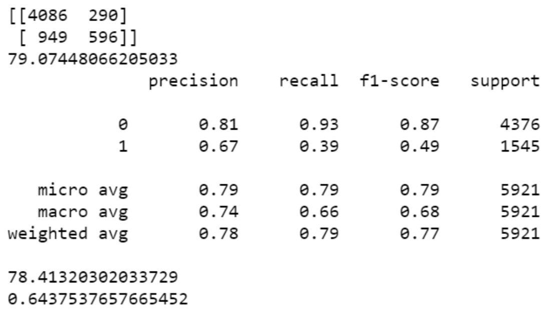
```
parameters = [{'criterion': ['gini'],'min_samples_leaf':[5,10,20,50,100],'max_depth':[5,10,20,50,100]},
              {'criterion': ['entropy'],'min_samples_leaf':[5,10,20,50,100],'max_depth':[5,10,20,50,100]}]
```

And the best combination of parameter is {'criterion': 'gini', 'max_depth': 50, 'min_samples_leaf': 5} which reports an accuracy of 85.75% with 10-fold cross validation.

3. Boosting
I have used Ada Boost algorithm to perform the Boosting. I choose this as it performed better then the Gradient Boosting classifier.
The results obtained from Gradient Boost is

```
[[4376    0]
 [1511   34]]
74.48066205032934
              precision    recall  f1-score   support

           0       0.74      1.00      0.85      4376
           1       1.00      0.02      0.04      1545

   micro avg       0.74      0.74      0.74      5921
   macro avg       0.87      0.51      0.45      5921
weighted avg       0.81      0.74      0.64      5921

74.06982906095448
0.21323727016397767
```

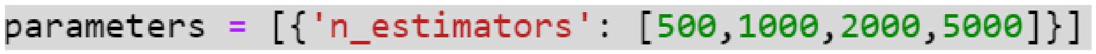
The mean accuracy with 10-fold cross validation is 74.06% with a standard deviation of 0.2%.
The Result from Ada Boost is



```
[[4086  290]
 [ 949  596]]
79.07448066205033
              precision    recall  f1-score   support

           0       0.81      0.93      0.87      4376
           1       0.67      0.39      0.49      1545

   micro avg       0.79      0.79      0.79      5921
   macro avg       0.74      0.66      0.68      5921
weighted avg       0.78      0.79      0.77      5921

78.41320302033729
0.6437537657665452
```

The mean accuracy with 10-fold cross validation is 78.41% with a standard deviation of 0.6%.

- Experiment with pruning in Boosting
  I have experimented with number of estimators used in the boosting by grid search and I am reporting the best performing combination of parameter values.



```
parameters = [{'n_estimators': [500,1000,2000,5000]}]
```

The best performing boosting has number of estimators as 500 and its accuracy score is around 83.64%

The best performing algorithm was a Decision tree with a accuracy of around 86%. But in reality an Ensemble method would out do it as an Ensemble method will have a better Bias Variance Trade off.

Part-2

In Part-2, I have tried to predict if a customer is going to quit the services of a Telecom operator based on various features.
I selected this dataset as Churn is a one of the biggest problems in the telecom industry. Research has shown that the average monthly churn rate among the top 4 wireless carriers in the US is 1.9% - 2%. If we are able to predict the churn of a customer this will be worth a lot of money as Telecom company can try to save the customer who may quit their services by offering him better plans and offers.

To better understand this problem we can define Churn as Customer attrition, also known as customer churn, customer turnover, or customer defection, is the loss of clients or customers.

Telephone service companies, Internet service providers, pay TV companies, insurance firms, and alarm monitoring services, often use customer attrition analysis and customer attrition rates as one of their key business metrics because the cost of retaining an existing customer is far less than acquiring a new one. Companies from these sectors often have customer service branches which attempt to win back defecting clients, because recovered long-term customers can be worth much more to a company than newly recruited clients.

Companies usually make a distinction between voluntary churn and involuntary churn. Voluntary churn occurs due to a decision by the customer to switch to another company or service provider, involuntary churn occurs due to circumstances such as a customer's relocation to a long-term care facility, death, or the relocation to a distant location. In most applications, involuntary reasons for churn are excluded from the analytical models. Analysts tend to concentrate on voluntary churn, because it typically occurs due to factors of the company-customer relationship which companies control, such as how billing interactions are handled or how after-sales help is provided.

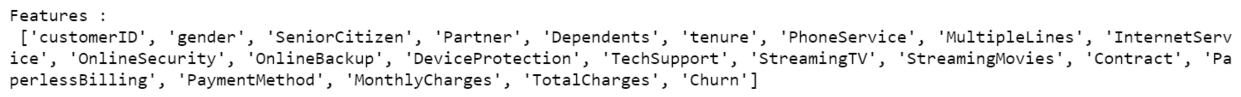
predictive analytics use churn prediction models that predict customer churn by assessing their propensity of risk to churn. Since these models generate a small prioritized list of potential defectors, they are effective at focusing customer retention marketing programs on the subset of the customer base who are most vulnerable to churn.

Summary statistics of the dataset used:

• Dataset consists of 7043 observations on 21 variables.

• The dependent variable for the Classification model is Churn variable which is Binary column with values "Yes" if the customer quit the services and "No" if the customer is still with the telecom operator.

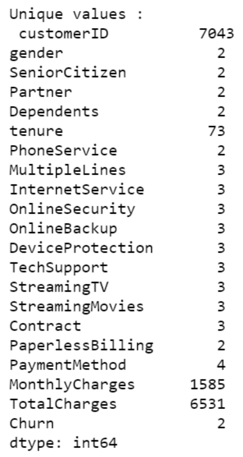
The Features are:



```
Features :
 ['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure', 'PhoneService', 'MultipleLines', 'InternetServ
ice', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract', 'Pa
perlessBilling', 'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn']
```

There are No missing values in the dataset.

The Unique values in the dataset are



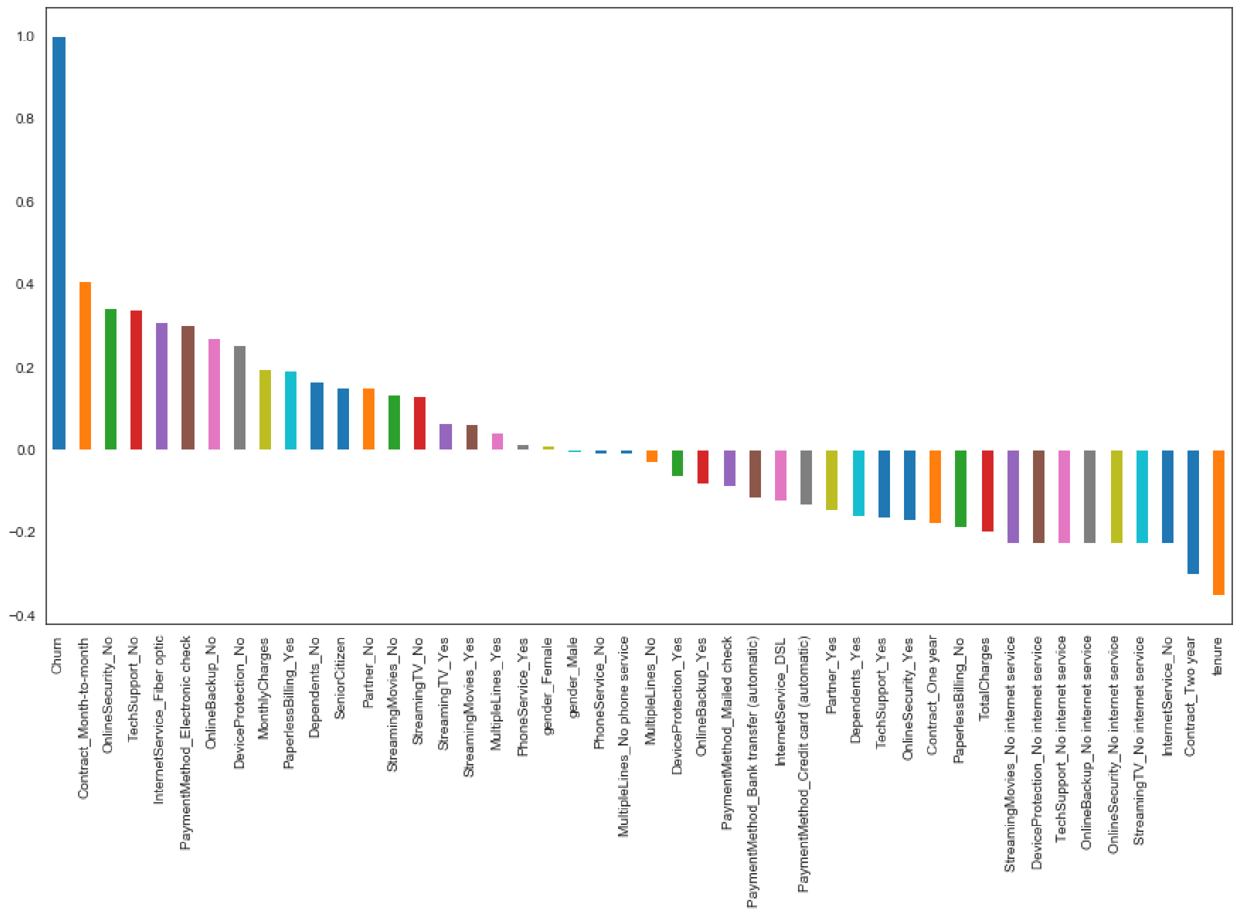
```
Unique values :
 customerID         7043
gender                2
SeniorCitizen         2
Partner               2
Dependents            2
tenure               73
PhoneService          2
MultipleLines         3
InternetService       3
OnlineSecurity        3
OnlineBackup          3
DeviceProtection      3
TechSupport           3
StreamingTV           3
StreamingMovies       3
Contract              3
PaperlessBilling      2
PaymentMethod         4
MonthlyCharges     1585
TotalCharges       6531
Churn                 2
dtype: int64
```

We find out there are 11 NA's in the Total charges columns So, we drop these observations and we Remove customer IDs from the data set as this feature is not useful for our classification.

We then proceed to convert the predictor variable in a binary numeric variable with values 1 for "Yes" and 0 for "No". And as Machine Algorithms only understand numbers we convert all the categorical variables into dummy variables.

We now have 46 columns.

The Correlation of "Churn" with other variables:



From the above creations we get to know Month to month contracts, absence of online security and tech support seem to be positively correlated with churn. While, tenure, two-year contracts seem to be negatively correlated with churn.

Interestingly, services such as Online security, streaming TV, online backup, tech support, etc. without internet connection seem to be negatively related to churn

I have split the dataset into training set and testing set with a ratio of 30% testing set and 70% training set. And standard student scaling is done on all the feature variables. Student scaled variable has a mean of 0 and standard distribution of 1.

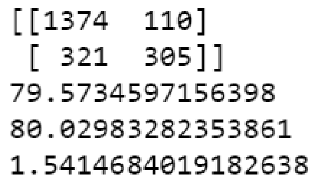
Fitting classification Algorithms to the dataset and checking which performs the best

1. Support Vector Classification
   - Linear SVC

```
[[1340  144]
 [ 289  337]]
79.478672985782
79.94816146599476
1.4313524902913912
```

From the confusion matrix above we see that there is a lot of misclassification and mean accuracy reported after performing a 10 fold cross validation is 79.94% and a standard deviation of 1.43%, which is okay but let's see the performance of the non linear SVM's

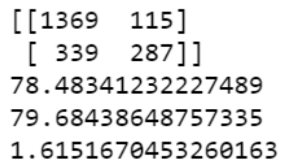
- RBF kernel



```
[[1374  110]
 [ 321  305]]
79.5734597156398
80.02983282353861
1.5414684019182638
```

The RBF has a very minute difference compared with the Linear SVM, mean accuracy reported after performing a 10 fold cross validation is 80% and a standard deviation of 1.54% which is not that good.

- Polynomial Kernel

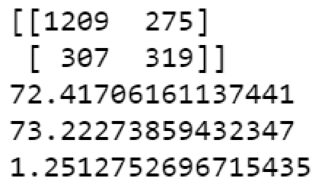


```
[[1369  115]
 [ 339  287]]
78.48341232227489
79.68438648757335
1.6151670453260163
```

This actually performs a bit poorer compared to the RBF but just a bit, , mean accuracy reported after performing a 10 fold cross validation is 79.68% and a standard deviation of 1.61% which is not that good.

2. Decision trees

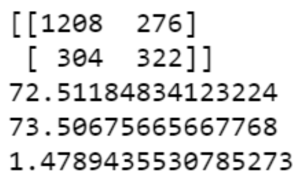
- Information gain as the criteria for splitting the nodes



```
[[1209  275]
 [ 307  319]]
72.41706161137441
73.22273859432347
1.2512752696715435
```

When we use information gain which is also called "entropy"as the splitting criteria we get the mean accuracy as around 73.22% and a standard deviation of 1.25% with 10 folds cross validation
which is not as good as a Kernel RBF SVM which was the best performing SVM.

- Gini Index as a criterion for splitting the nodes



```
[[1208  276]
 [ 304  322]]
72.51184834123224
73.50675665667768
1.4789435530785273
```

When we use the Gini index as criterion for splitting the nodes, we get the mean accuracy as around 73.50% and a standard deviation of 1.47% with 10 folds cross validation
which is just a bit higher than a using entropy decision tree. Therefore, we can conclude that there is not much difference when we use either entropy or

Gini Index. But both of them are not as good as a kernel RBF SVM on this dataset.

- Experimenting with pruning the decision tree

  As Grid Search technique provides a easy way to know which is the best combination of the parameters., when I experimented with pruning. The various values that I have passed for the parameters are as follows.

```
parameters = [{'criterion': ['gini'],'min_samples_leaf':[5,10,20,50,100],'max_depth':[5,10,20,50,100]},
              {'criterion': ['entropy'],'min_samples_leaf':[5,10,20,50,100],'max_depth':[5,10,20,50,100]}]
```

  The best performing tree has the parameter values of

  {'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 100}

  And has an accuracy of around 80% which shows that pruning considerably improves the performance of the decision tree.

3 Boosting

I have used Ada Boost algorithm to perform the Boosting. I choose this as it performed better than the Gradient Boosting classifier.
The results obtained from Gradient Boosting is

```
[[4376    0]
 [1511   34]]
74.48066205032934
74.06982906095448
0.21323727016397767
```

The mean accuracy obtained is 74.06% with a standard deviation of around 0.2%
The result from Ada Boosting is

```
[[4086  290]
 [ 949  596]]
79.07448066205033
78.41320302033729
0.6437537657665452
```

This shows that when we use Ada Boost it performs better then Gradient boosting algorithm considerably, It has an accuracy of 78.41 % when 10 fold cross validation is used with a standard deviation of 0.6% which signals the 10 accuracies are tightly packed.

- Experimenting with Pruning in Boosting

  I have experimented with number of estimators used in the boosting by grid search and I am reporting the best performing combination of parameter values.

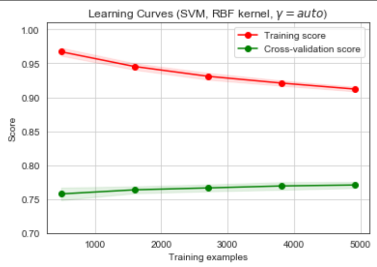
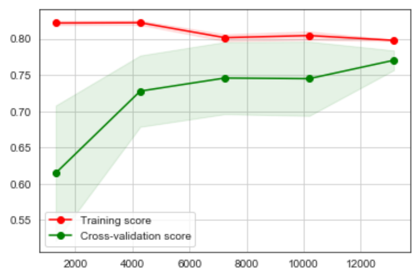
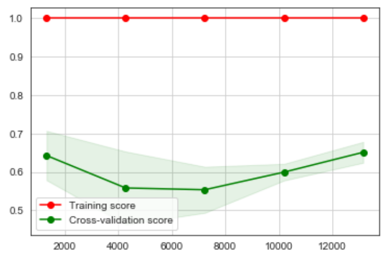
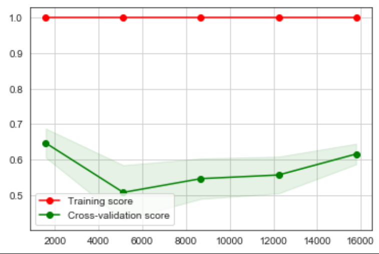
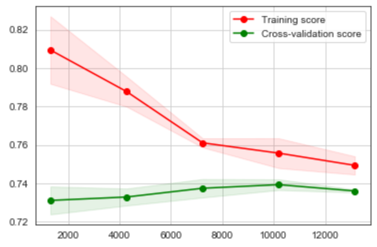
```
parameters = [{'n_estimators': [50,100,200,300,500,1000,1500]}]
```

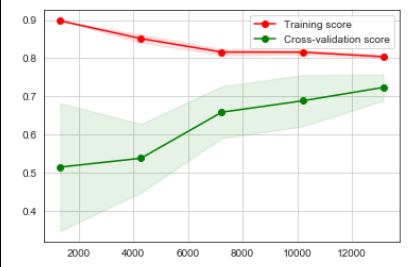
The best performing boosting has number of estimators as 500 and its accuracy score is around 80.13%.

The best performing algorithm was a Ensemble Boosting method with number of estimators as 500 and an accuracy around 80%

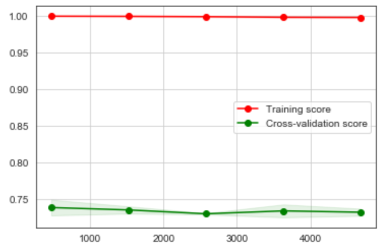
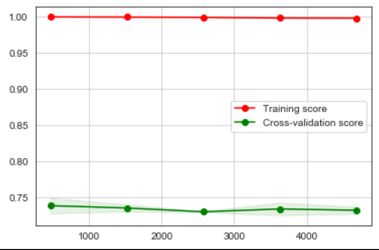
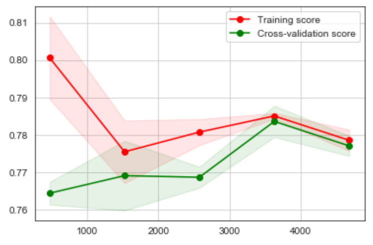
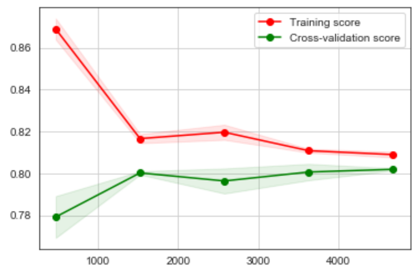
The learning curves

Part-1

Algorithm	**Learning curve**	**Explanation**
Linear SVM		The learning curve shows that the training score is coming down as the training size increases and also shows that testing or the CV score is slowly increasing with the increase in Training size.
RBF kernel SVM		In this curve the optimal size of the training set is almost reached at 13000
Gini Decision tree		The error rate is stuck at 1 for all training set sizes and cross validation error increases in U shape
Entropy Decision Tree		The error rate is stuck at 1 for all training set sizes and cross validation error increases in U shape
Gradient Boosting		The train error and the test error converge at around 13000 Training size which is the best Bias Variance trade off

Ada Boost		The train error and the test error converge at around 13000 Training size which is the best Bias Variance trade off

Part-2

Algorithm	**Learning curve**	**Explanation**
Gini Decision tree		The training error is 1 regardless of the training set size. The Decision tree is not a good algorithm as it is over fitting the data in this case
Entropy Decision Tree		The training error is 1 regardless of the training set size. The Decision tree is not a good algorithm as it is over fitting the data in this case
Gradient Boosting		Gradient Boosting brings down the Training error rate and also the testing error rises and then drops down again after around 3800 training observations this might be a optimal number of training observations
Ada Boost		Ada Boosting brings down the Training error rate and also the testing error rises and almost converges at around 4750 training set size which might be a optimal size for training set

Linear SVM and RBF kernel SVM Learning curves could not be generated for Part-2 as the system was crashing repeatedly when I was trying generate it.(It might be a problem with my computer)