Prepared by: Eeshan Phatak **Date: 03/09/2020**

BUAN 6341
APPLIED MACHINE LEARNING
ASSIGNMENT 2
SVM, Decision Tree and Boosting Algorithms Implementation

Report Summary

- Prepared Two Datasets (Sgemm Product Dataset and Bank Dataset) for classification
- Implemented SVM, Decision Tree and Boosting Classification algorithms to calculate different performance metrics on the datasets
- Implemented SVM Classification with 3 different Kernels
- Implemented Decision Tree Classification with Pruning and different Criterions
- Implemented AdaBoost Classification with Pruning and Changing number of Estimators

Data Description

- Dataset 1 is the dataset of factors affecting the GPU Kernel Runtime. This dataset was also used in the assignment 1 for Linear and Logistic Regression. The data has 241600 rows and 18 columns out of which last 4 columns (dependent variable columns) have been combined into one for classification purposes.
- Dataset 2 is a bank client dataset and the dependent variable is y (Has the client subscribed a term deposit? Y/N). This data has 20 features and 1 dependent binary variable. The features include data regarding to clients which help find out if the client will subscribe to a deposit or not.

Data Preparation

Dataset 1

- Took average of 4 RunTime columns to get 1 single dependent variable Run_Avg
- In dataset 1 the values of the dependent variable (Run_Avg) is continuous which could not be used for classification
- Divided the column values into 1 and 0 by changing all the values equal and above the median value to 1 and other values to 0
- Normalized the independent variables in the dataset to get better classification results

Dataset 2

- Converted Text columns including the dependent variable to numerical values
- Applied filters to convert unknown values to known values
- Dependent variable y contained ‘Yes’ and ‘No’ values so converted them to 1 and 0
- Normalized the independent variables in the dataset to get better classification results

Prepared by: Eeshan Phatak **Date: 03/09/2020**

SVM Classification

- Calculated confusion matrices and classification reports for all the 3 kernels
- Calculated MSE and Accuracy for 3 different kernels in both the datasets

Dataset 1

Implemented SVM classification for following factors:

Without Cross validation and kernel = Linear

You will find execution of all the kernels in the code. I have pasted screenshot of only one of the kernels.

```
svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, y_train)
y_pred = svclassifier.predict(X_test)
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[31304  4949]
 [ 7276 28951]]
              precision    recall  f1-score   support

           0       0.81      0.86      0.84     36253
           1       0.85      0.80      0.83     36227

    accuracy                           0.83     72480
   macro avg       0.83      0.83      0.83     72480
weighted avg       0.83      0.83      0.83     72480
```

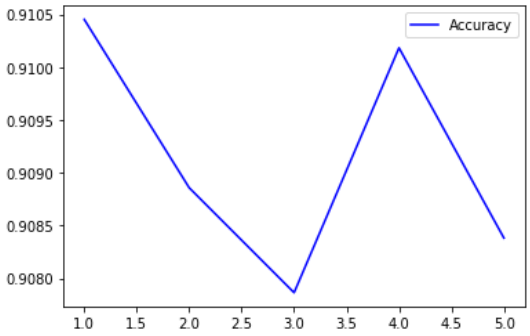
With Cross Validation and Kernels Linear, Poly and rbf

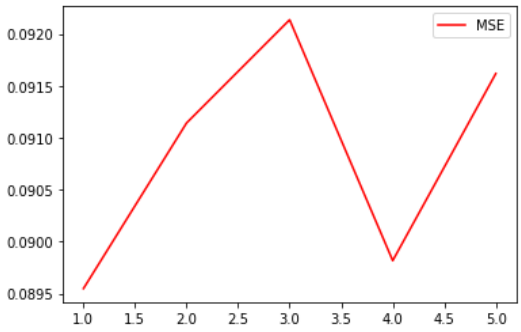
- I have written code for all three kernels and attached the plots for rbf kernel below.
- I also tried testing my accuracy and error by keeping shuffle as True or False in the K-Fold Validation.
- I deduced that when shuffle is True the records are mixed rather than being grouped and it gives a better accuracy.

5-fold Cross Validation with shuffle = True and SVM Kernel = poly

- Below are the plots of accuracy and error of the model with respect to the folds (1-5 for 5 fold)
- Below the plots is the image showing the average MSE and average accuracy of the 5-fold Cross Validation SVM model

Prepared by: Eeshan Phatak **Date: 03/09/2020**






```
avg_mse_sv=mean(mse)
print(avg_mse_sv)
avg_accuracy_sv=mean(sv_accuracy)
print(avg_accuracy_sv)
```

```
0.09085264900662252
0.9091473509933775
```

Dataset 2

Implemented SVM classification for following factors:

Without Cross validation and kernel = Poly

You will find execution of all the kernels in the code. I have pasted screenshot of only one of the kernels.

```
svclassifier1 = SVC(kernel='poly',gamma='auto')
svclassifier1.fit(X_train, y_train)
y_pred1 = svclassifier1.predict(X_test)
print(confusion_matrix(y_test,y_pred1))
print(classification_report(y_test,y_pred1))
```

```
[[10712   220]
 [  905   520]]
              precision    recall  f1-score   support

           0       0.92      0.98      0.95     10932
           1       0.70      0.36      0.48      1425

    accuracy                           0.91     12357
   macro avg       0.81      0.67      0.72     12357
weighted avg       0.90      0.91      0.90     12357
```

Prepared by: Eeshan Phatak **Date: 03/09/2020**

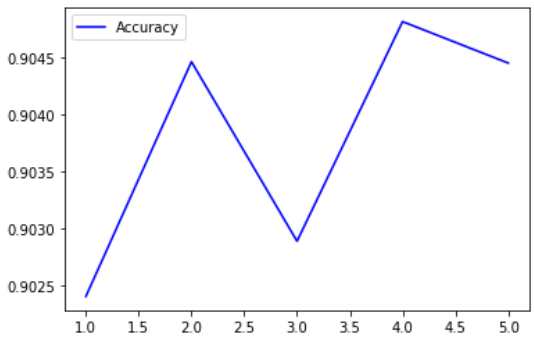
With Cross Validation and Kernels Linear, Poly and rbf

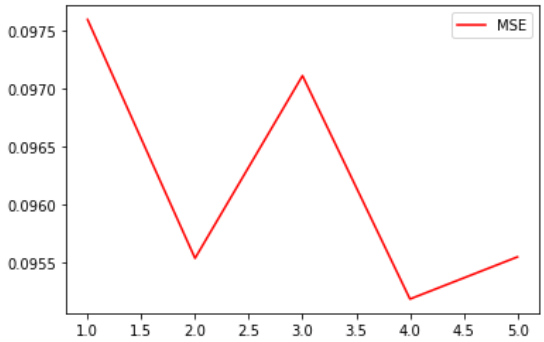
- I have written code for all three kernels and attached the plots for Linear and poly kernels below.
- I got the high accuracy rates ranging between 85-95 for all the kernels.
- I have attached plots for 2 of the kernels that I have executed in the code. You will be able to view plots for all the 3 kernels in the code.

5-fold Cross Validation with shuffle = True and SVM Kernel = linear and rbf

- Below are the plots or accuracy and error of the model with respect to the folds (1-5 for 5 fold)
- Below the plots is the image showing the average MSE and average accuracy of the 5-fold Cross Validation SVM model.

Linear Kernel



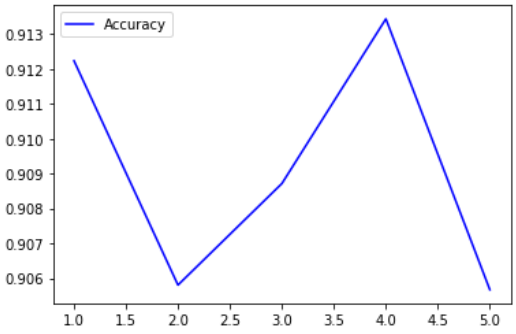


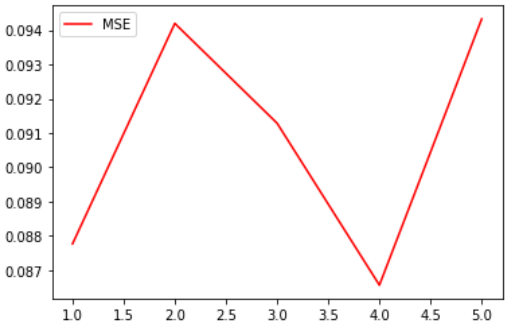

```
avg_mse_sv=mean(mse)
print(avg_mse_sv)
avg_accuracy_sv=mean(sv_accuracy)
print(avg_accuracy_sv)
```

```
0.09619302560763386
0.9038069743923661
```

Prepared by: Eeshan Phatak **Date: 03/09/2020**

RBF Kernel






```
avg_mse_sv=mean(mse)
print(avg_mse_sv)
avg_accuracy_sv=mean(sv_accuracy)
print(avg_accuracy_sv)
```

```
0.09082740692161032
0.9091725930783897
```

Decision Tree Classification

- Calculated classification reports and confusion matrix for the pruned and non-pruned decision trees
- Calculated Accuracy and MSE for decision trees with criterion gini and entropy and for pruned trees with levels from 2 to 20

Dataset 1

Implemented Decision Tree classification for following factors:

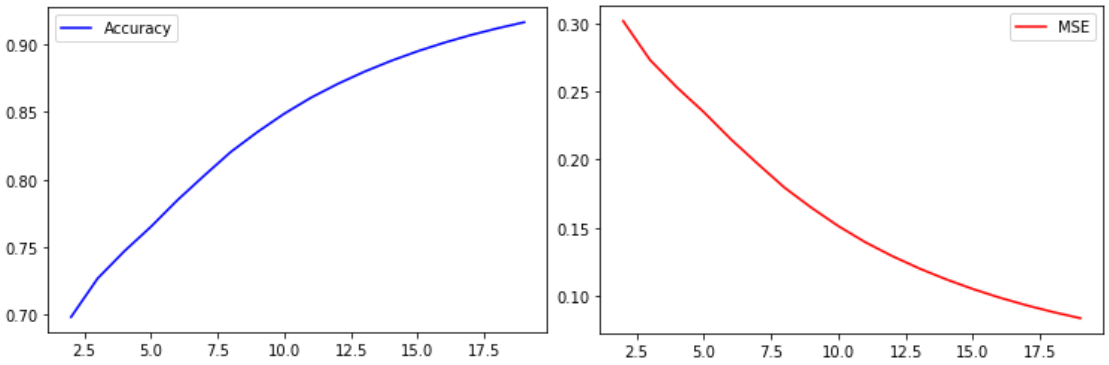
Without Cross Validation for Pruned and Non-Pruned Decision Trees

Below is Screenshot of Classification report and confusion matrix, Pruned with Max Depth = 7

```
dtclassifier1 = DecisionTreeClassifier(max_depth=7)
dtclassifier1.fit(X_train, y_train)
y_pred_dt1 = dtclassifier1.predict(X_test)
print(confusion_matrix(y_test,y_pred_dt1))
print(classification_report(y_test,y_pred_dt1))
```

```
[[33418  2842]
 [ 4890 31330]]
              precision    recall  f1-score   support

           0       0.87      0.92      0.90     36260
           1       0.92      0.86      0.89     36220

    accuracy                           0.89     72480
   macro avg       0.89      0.89      0.89     72480
weighted avg       0.89      0.89      0.89     72480
```

5-fold Cross Validation with shuffle = False and Decision Tree Criterion = gini with respect to 2-20 branches pruning



Dataset 2

Implemented Decision Tree classification for following factors:

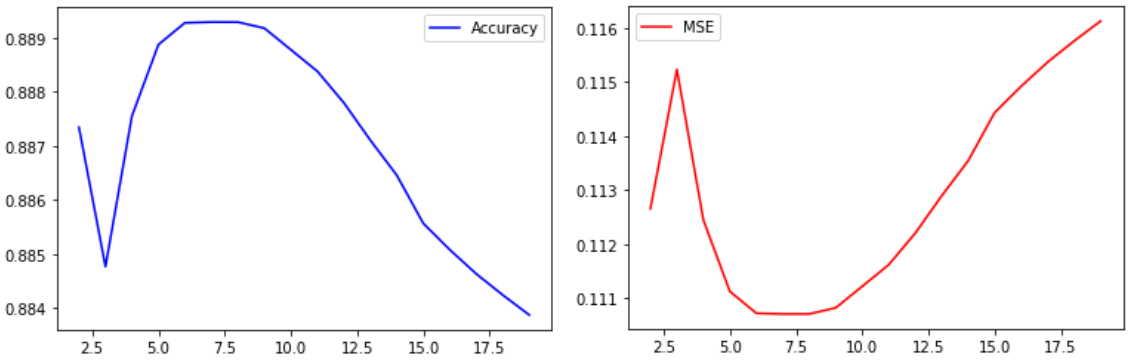
Without Cross Validation for Pruned and Non-Pruned Decision Trees

Below is Screenshot of Classification report and confusion matrix, Non-Pruned

```
dtclassifier = DecisionTreeClassifier()
dtclassifier.fit(X_train, y_train)
y_pred_dt = dtclassifier.predict(X_test)
print(confusion_matrix(y_test,y_pred_dt))
print(classification_report(y_test,y_pred_dt))
```

```
[[10270   662]
 [  693   732]]
              precision    recall  f1-score   support

           0       0.94      0.94      0.94     10932
           1       0.53      0.51      0.52      1425

    accuracy                           0.89     12357
   macro avg       0.73      0.73      0.73     12357
weighted avg       0.89      0.89      0.89     12357
```

5-fold Cross Validation with shuffle = False and Decision Tree Criterion = entropy with respect to 2-20 branches pruning. The accuracy seems to be the maximum when the Max Depth =7.



Boosting Decision Trees for Classification

- Calculated classification reports and confusion matrix for different number of trees and max depths for Boosting
- I have used AdaBoost with different number of trees and different max depths

Dataset 1

Implemented Boosting for the following factors:

Without K-Fold Validation

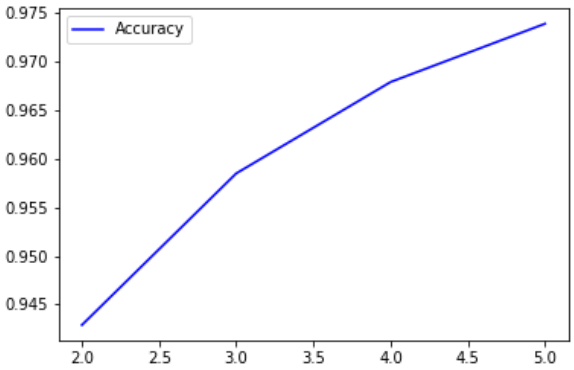
Below is a screenshot of Boosting using 300 estimators and max depth =

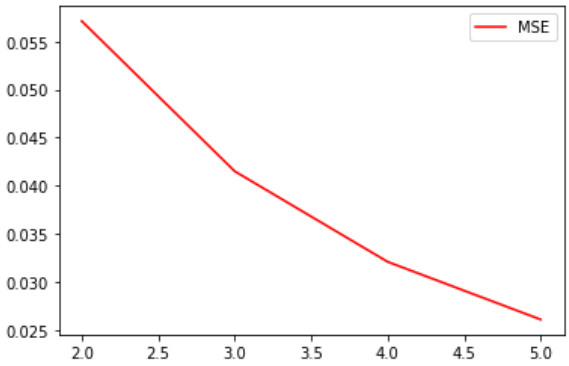
```
boost_classifier = AdaBoostClassifier(DecisionTreeClassifier(max_depth=2), n_estimators=400, random_state=None)
boost_classifier.fit(X_train, y_train)
y_pred_boost = boost_classifier.predict(X_test)
print(confusion_matrix(y_test,y_pred_boost))
print(classification_report(y_test,y_pred_boost))
```

```
[[22875  1233]
 [ 1527 22685]]
              precision    recall  f1-score   support

           0       0.94      0.95      0.94     24108
           1       0.95      0.94      0.94     24212

    accuracy                           0.94     48320
   macro avg       0.94      0.94      0.94     48320
weighted avg       0.94      0.94      0.94     48320
```

With 5-Fold Validation and shuffle = True 5. Pruned from 2-5 levels.

For this data the accuracy of the model increases if we prune at 4th or 5th level.





Dataset 2

Implemented Boosting for the following factors:

Without K-Fold Validation

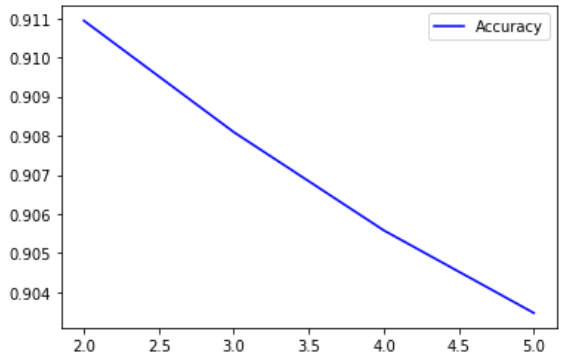
Below is a screenshot of Boosting using 300 estimators and max depth =3

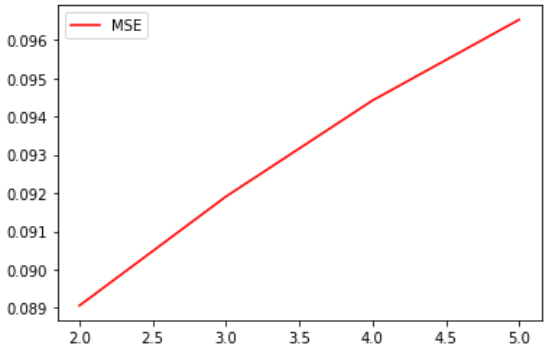
```
boost_classifier = AdaBoostClassifier(DecisionTreeClassifier(max_depth=3), n_estimators=300, random_state=None)
boost_classifier.fit(X_train, y_train)
y_pred_boost = boost_classifier.predict(X_test)
print(confusion_matrix(y_test,y_pred_boost))
print(classification_report(y_test,y_pred_boost))
```

```
[[10446   486]
 [  673   752]]
              precision    recall  f1-score   support

           0       0.94      0.96      0.95     10932
           1       0.61      0.53      0.56      1425

    accuracy                           0.91     12357
   macro avg       0.77      0.74      0.76     12357
weighted avg       0.90      0.91      0.90     12357
```

With K-Fold Validation with shuffle=True

The below plots are of accuracy and MSE for different pruning from 2-5 branches and with estimators=300. With more pruning the Accuracy reduces.

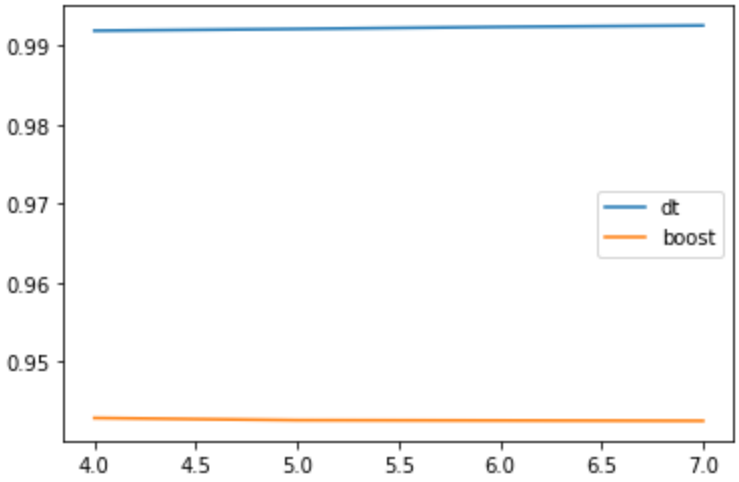




Comparison between SVM, Decision Tree and Boosting

- Decision tree gives the most accuracy and is the fastest in runtime.
- Boosting gives a good accuracy and takes moderate amount of time to run.
- SVM is very slow to run. It gives good accuracy and less error, but it takes much longer than other algorithms. The RBF kernel gave the better accuracies than the other 2 kernels. For the second dataset RBF kernel gives the highest classification accuracy more than boosting and decision trees.
- For the following data I would say that Decision Tree is a very good classification algorithm as it takes very little time and gives a very good accuracy

Prepared by: Eeshan Phatak **Date: 03/09/2020**



Above is a plot of accuracies of boosting and Decision tree classifiers for K-Fold validations from 4fold to 7fold validation. The decision tree without pruning gives very high accuracies while boosting gives a good accuracy but less than Decision trees.

Observations made through the assignment

- SVM is not a suggested algorithm for the first dataset
- All the algorithms gave almost similar results for the second dataset in terms of time and accuracy both.
- Using different kernels helps achieve better results (less error)
- In case of the first dataset pruning is not required in Decision Trees as it reduces the accuracy of the model.
- Entropy and Gini Criterions do not seem to have a large effect on the accuracy of the prediction models.
- Increasing the number of estimators in the boosting algorithm helps increase the accuracy
- Max Depth is a good measure to calculate where to prune the data.