Case study of the Fused Model and AdaBoost Ensemble with Decision Tree as the base learner.

Oksana Dura
Student ID: 1316268

May 10, 2023

Abstract

Every day hundreds and thousands of messages are being sent all over the world. Some of them are important messages while the rest is marketing or even scam messages. In our phones or emails, our messages are classified for us. In this case study we will learn two methods on how these classifications are made and most important we will be able to study different classification methods and find out their accuracy for a given dataset. In this report, we will be classifying email messages as spam or ham using Decision Tree Classification and Random Forest Classification and studying their classification accuracies.

1 Introduction

For the case study, we were given a dataset containing 57 attributes that encode the total number a word or character occurs, and a total of 4601 instances. The dataset classifies email messages as spam or ham. In the given dataset we had to fuse three classifiers using the majority voting rule: (1) Decision Tree, (2) Gaussian Naïve Bayes, and (3) Logistic Regression. Then compare the accuracy of the fused model with (4) AdaBoost Ensemble with Decision Trees as the base learner, and (5) Random Forests.

- Compare the accuracies of the fused model with AdaBoost Ensemble with Decision Tree as the base learner. Train the classifiers using the first 1000 instances and use the remaining 3601 for testing.
- Compare the accuracies of the fused model with Random Forest (with 1000 base learners). Train the classifiers using the first 1000 instances and use the remaining 3601 for testing.
- Study the impact of training sample size on the accuracies of the fused classifier and the AdaBoost Ensemble with Decision Tree as the base learner. Compare their accuracies with the following training-test splits: 50%-50%, 60%-40%, 70%-30%, and 80%-20%.

2 Understanding the data

2.1 Optimizing the parameters for the Linear Regression

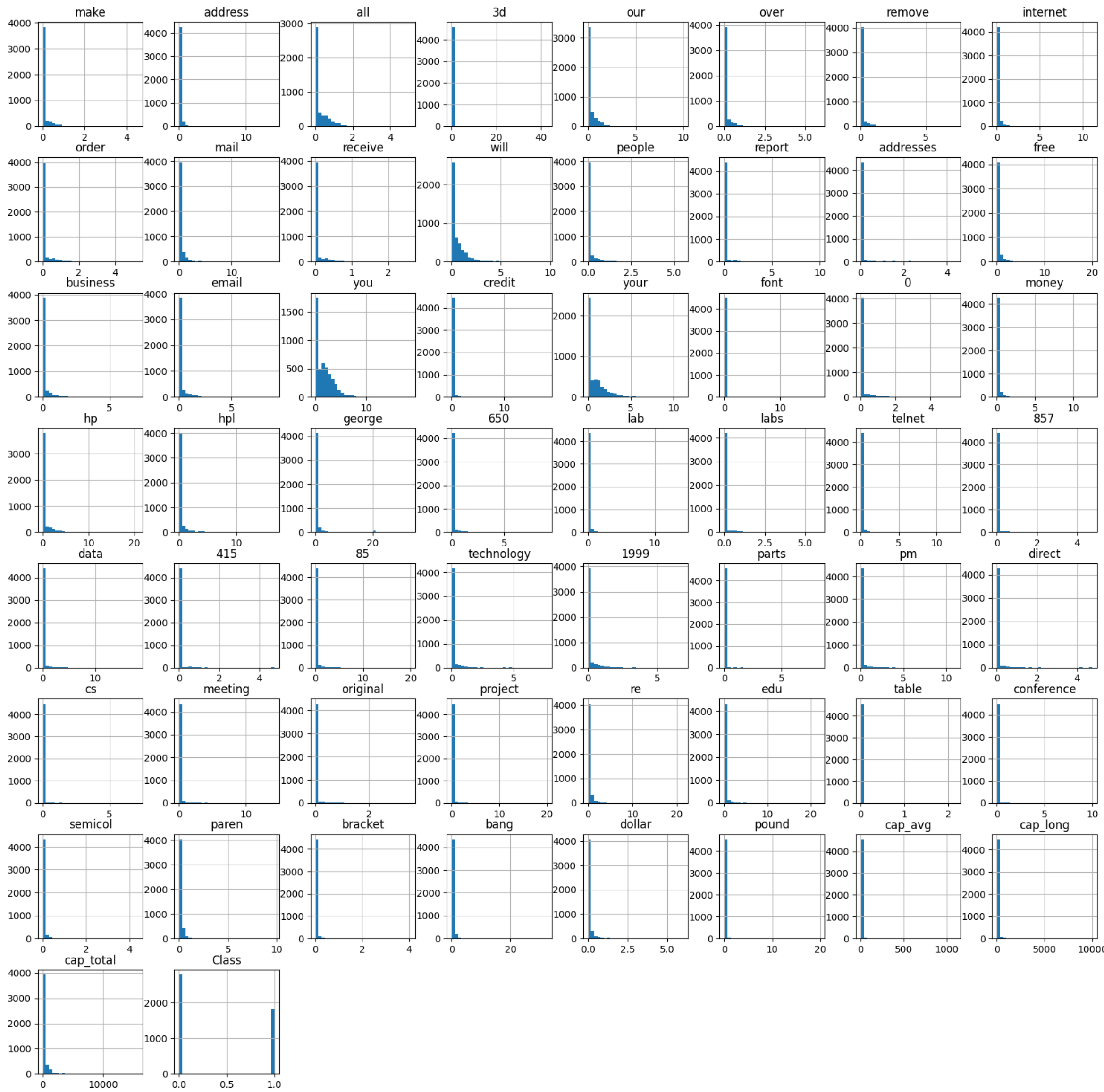
Before implementing the classification methods is a good practice to understand the data you are working with, and make the necessary changes to get the best output. Some of the changes may include cleaning it, checking for missing data, replacing some values, and naming the features. All of the preparations of the data depend on the dataset that we were given.

Important elements from the dataset:

- The Dataset consists of 57 features 4601 samples.
- The data consists of 55 float type, 2 int type, and 1 object type.
- The object type named "Class" takes two values, either ham or spam.
- There are no missing values in the dataset.

2.2 Visualizing the dataset

The figure below represents per feature histogram



I have assigned 1 to 'spam' and 0 to ham in the dataset because it makes it easier for me to work with having them as integers. The replacement is seen in the following code:

```
[5] spam_dataset_dataframe['Class']=spam_dataset_dataframe['Class'].apply(lambda x: 1 if x=='spam' else 0)
    spam_dataset_dataframe.head()
```

Here we can see the changes that happened to the "Class" column in the dataset. :

	make	address	all	3d	our	over	remove	internet	order	mail	...	semicol	paren	bracket	bang	dollar	pound	cap_avg	cap_long	cap_total	Class
0	0.00	0.00	0.29	0.0	0.00	0.00	0.00	0.00	0.00	0.00	...	0.000	0.178	0.0	0.044	0.000	0.00	1.666	10	180	0
1	0.46	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00	0.00	...	0.000	0.125	0.0	0.000	0.000	0.00	1.510	10	74	0
2	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00	0.00	...	0.000	0.000	0.0	0.000	0.000	0.00	1.718	11	55	0
3	0.33	0.44	0.37	0.0	0.14	0.11	0.00	0.07	0.97	1.16	...	0.006	0.159	0.0	0.069	0.221	0.11	3.426	72	819	1
4	0.00	2.08	0.00	0.0	3.12	0.00	1.04	0.00	0.00	0.00	...	0.000	0.000	0.0	0.263	0.000	0.00	1.428	4	20	1

5 rows × 58 columns

2.3 Splitting the dataset

The dataset was split for training and testing, the respective variables were the following: **spam_training_set, spam_test_set**. The training set contains 1000 instances and the testing set contains 3501 instances. The following figures show the implementation of the slipt method.

```
[8] #Create a training and test set
    spam_training_set, spam_test_set = train_test_split(spam_dataset_dataframe, test_size = 3601
    , random_state = 42)
    spam_dataset_dataframe.keys()
```

```
spam_training_data, spam_training_target = spam_training_set[["make", "address", "all","3d","our", "over", "remove", "internet", "order", "mail", "receive", "will",
spam_test_data, spam_test_target = spam_test_set[["make", "address", "all","3d","our", "over", "remove", "internet", "order", "mail", "receive", "will", "people", "
spam_training_data.head()
```

3 First Task

3.1 Compare the accuracies of the fused model with the AdaBoost Ensemble with Decision Tree as the base learner.

3.2 Decision Tree

For the decision tree, we use the parameters from the previous assignments that gave the highest accuracy. After the program is run we see the following parameters produce the highest accuracy:

- random_state = 101 (Is constant)
- criterion = entropy
- max_depth = 12
- max_features = None
- splitter = best

3.3 Optimizing the parameters for the Liner Regression

Linear Regression has a multitude of parameters that can be changed and adapted to give the best output. For this assignment, we are going to concentrate only on a few of them, I am doing parameter optimization in order to get the best parameters for the classifier that is going to be used for the Voting Classifier.

- The first parameter is penalty_LR which specifies the norm of the penalty:
  - None: no penalty is added;
  - 'l2': add a L2 penalty term and it is the default choice;
  - 'l1': add a L1 penalty term;
  - 'elasticnet': both L1 and L2 penalty terms are added..

  Since we are using the default solver = 'lbfgs', we can only take into consideration two of the parameters. For this purpose, we will consider the following list penalty_LR = ["l2", None]. We are going to be iterating through all of the elements of the list penalty_LR compare the accuracies and find the optimal value for penalty_LR.

- The second parameter that is going to be considered is multi_class. There are three possible choices here, multi_class_LR = ["auto","ovr", "multinomial"]. We are going to be iterating through each of them, comparing the accuracies, and finding the highest accuracy.

- There is a third parameter that is considered for the Logistic Regression classifier, and that is max_iter_LR = [10000,100000]. Maximum number of iterations taken for the solvers to converge. The default number of iterations is 100, but in this case, that was not enough. In all of the classifiers, I have assigned random_state = 101.

The following code shows the implementation of the optimization of parameters

```
highest_accuracy = 0
penalty_LR = [ "l2",  None]
multi_class_LR = ["auto","ovr", "multinomial"]
max_iter_LR = [ 10000,100000]
u = ""
d = " "
f = 0

for g in penalty_LR:
  for h in multi_class_LR:
    for k in max_iter_LR:
      clf_lr = LogisticRegression(penalty = g, multi_class = h, max_iter = k,  random_state = 101)
      clf_lr.fit(spam_training_data,spam_training_target)
      spam_test_target_predict=clf_lr.predict(spam_test_data)
      print("For penalty = ",g,", and multi_class = ", h ,"and max_iter: ",k, "the accuracy score is: ", accuracy_score(spam_test_target,spam_test_target_predict))

      if accuracy_score(spam_test_target,spam_test_target_predict) > highest_accuracy:
          highest_accuracy = accuracy_score(spam_test_target,spam_test_target_predict)
          u = g
          d = h
          f = k


print("The random forest with the highest accuracy ",highest_accuracy, "has the following parameters: penalty = ", u, " and multi_class = ", d, " and max_iteration =
```

```
clf_lr = LogisticRegression(penalty = u ,multi_class = d , max_iter= f , random_state = 101)
clf_lr.fit(spam_training_data,spam_training_target)
spam_test_target_predict=clf_lr.predict(spam_test_data)
c_m_lr = confusion_matrix(spam_test_target,spam_test_target_predict)
c_r_lr = classification_report(spam_test_target,spam_test_target_predict)
a_s_lr = accuracy_score(spam_test_target,spam_test_target_predict)


# Compare observed value and Predicted value
print("Prediction for 20 observation:    ",clf_lr.predict(spam_test_data[0:20]))
print("Actual values for 20 observation: ",spam_test_target[0:20].values)
print(c_m_lr)
print(c_r_lr)
print(a_s_lr)
```

As we wrote above the parameter optimization is considering three of the parameters that Linear Regression has. For the first parameter penalty, we had 2 elements, and for the second feature multi_class there are 3 elements, for the third parameter we had 2 values. Hence to find to best accuracy we have to compare all the possible combinations, in this case, we get 12 accuracy values. We need to compare those and get the highest accuracy.

```
For penalty =  12 , and multi_class =  auto and max_iter:  10000 the accuracy score is:  0.9202999166898084
For penalty =  12 , and multi_class =  auto and max_iter:  100000 the accuracy score is:  0.9202999166898084
For penalty =  12 , and multi_class =  ovr and max_iter:  10000 the accuracy score is:  0.9202999166898084
For penalty =  12 , and multi_class =  ovr and max_iter:  100000 the accuracy score is:  0.9202999166898084
For penalty =  12 , and multi_class =  multinomial and max_iter:  10000 the accuracy score is:  0.9202999166898084
For penalty =  12 , and multi_class =  multinomial and max_iter:  100000 the accuracy score is:  0.9202999166898084
For penalty =  None , and multi_class =  auto and max_iter:  10000 the accuracy score is:  0.9128019994445987
For penalty =  None , and multi_class =  auto and max_iter:  100000 the accuracy score is:  0.9128019994445987
For penalty =  None , and multi_class =  ovr and max_iter:  10000 the accuracy score is:  0.9128019994445987
For penalty =  None , and multi_class =  ovr and max_iter:  100000 the accuracy score is:  0.9128019994445987
For penalty =  None , and multi_class =  multinomial and max_iter:  10000 the accuracy score is:  0.9136351013607331
For penalty =  None , and multi_class =  multinomial and max_iter:  100000 the accuracy score is:  0.9136351013607331
```

```
The random forest with the highest accuracy  0.9202999166898084 has the following parameters: penalty =  12  and multi_class =  auto  and max_iteration =  10000
```

```
Prediction for 20 observation:     [0 0 1 0 0 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1]
Actual values for 20 observation:  [0 0 1 0 0 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1]
```

```
[[2025  160]
 [ 127 1289]]
```

```
              precision    recall  f1-score   support

           0       0.94      0.93      0.93      2185
           1       0.89      0.91      0.90      1416

    accuracy                           0.92      3601
   macro avg       0.92      0.92      0.92      3601
weighted avg       0.92      0.92      0.92      3601
```

3.4 Creating the fused model

3.5 Fused Model

For the fused model, we are going to use three classifiers using the majority voting rule. The three classifiers are the following: (1) Decision Tree, (2) Gaussian Naïve Bayes, and (3) Logistic Regression. A voting classifier is a machine learning algorithm that combines the predictions of multiple individual models to make a final prediction. Each individual model in the ensemble votes on the predicted class label for a given input, and the voting classifier aggregates the votes to make a final decision. In this case, the voting classifier will be can be configured to use hard voting. In hard voting, the voting classifier predicts the class label that receives the most votes from the individual models.

```
clf_dt = DecisionTreeClassifier(criterion = "entropy", max_features = None, splitter = "best", random_state = 101,max_depth = 12 )
clf_gnb = GaussianNB()
eclf = VotingClassifier(estimators = [('DT', clf_dt),('LR', clf_lr), ('GNB', clf_gnb)], voting = 'hard')
eclf.fit(spam_training_data,spam_training_target)
spam_test_target_predict=eclf.predict(spam_test_data)
c_m_VC = confusion_matrix(spam_test_target,spam_test_target_predict)
c_r_VC = classification_report(spam_test_target,spam_test_target_predict)
a_s_VC = accuracy_score(spam_test_target,spam_test_target_predict)
```


```
clf_dt = DecisionTreeClassifier(criterion = "entropy", max_features = None, splitter = "best", random_state = 101,max_depth = 12 )
clf_gnb = GaussianNB()
eclf = VotingClassifier(estimators = [('DT', clf_dt),('LR', clf_lr), ('GNB', clf_gnb)], voting = 'hard')
eclf.fit(spam_training_data,spam_training_target)
spam_test_target_predict=eclf.predict(spam_test_data)
c_m_VC = confusion_matrix(spam_test_target,spam_test_target_predict)
c_r_VC = classification_report(spam_test_target,spam_test_target_predict)
a_s_VC = accuracy_score(spam_test_target,spam_test_target_predict)
```


3.6 AdaBoost Ensemble with Decision Tree

AdaBoost (Adaptive Boosting) is a popular ensemble learning method that can be used with many types of base models, including decision trees (such as in the case of the assignment). The basic idea behind AdaBoost is to iteratively train a sequence of weak learners (i.e., models that are only slightly better than random guessing) on weighted versions of the training data, with the aim of gradually improving the overall performance of the ensemble.

```
ada = AdaBoostClassifier(
           estimator=clf_dt,
           random_state=101)


ada.fit(spam_training_data,spam_training_target)


spam_test_target_predict=ada.predict(spam_test_data)
c_m1 = confusion_matrix(spam_test_target,spam_test_target_predict)
c_r1 = classification_report(spam_test_target,spam_test_target_predict)
a_s1 = accuracy_score(spam_test_target,spam_test_target_predict)
```

```
# Compare observed value and Predicted value
print("Prediction for 20 observation:    ",ada.predict(spam_test_data[0:20]))
print("Actual values for 20 observation: ",spam_test_target[0:20].values)
print(c_m1)
print(c_r1)
print(a_s1)
```

3.7 Fused Model vs. AdaBoost Ensemble

Outputting 20 predicted values for the Fused Model

```
Prediction for 20 observation:     [0 0 1 0 1 1 1 0 1 1 0 0 0 0 1 1 0 1 1 1]
Actual values for 20 observation:  [0 0 1 0 0 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1]
```

Outputting 20 predicted values for AdaBoost Ensemble

```
Prediction for 20 observation:     [0 0 1 0 1 1 1 0 1 1 0 0 0 0 1 1 0 0 0 1]
Actual values for 20 observation:  [0 0 1 0 0 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1]
```

The Fused Model Outputs the following: Here we need to remember that 0 represents "ham", and 1 represents "spam"

```
[[1999  186]
 [ 105 1311]]
              precision    recall  f1-score   support

           0       0.95      0.91      0.93      2185
           1       0.88      0.93      0.90      1416

    accuracy                           0.92      3601
   macro avg       0.91      0.92      0.92      3601
weighted avg       0.92      0.92      0.92      3601

0.9191891141349625
```

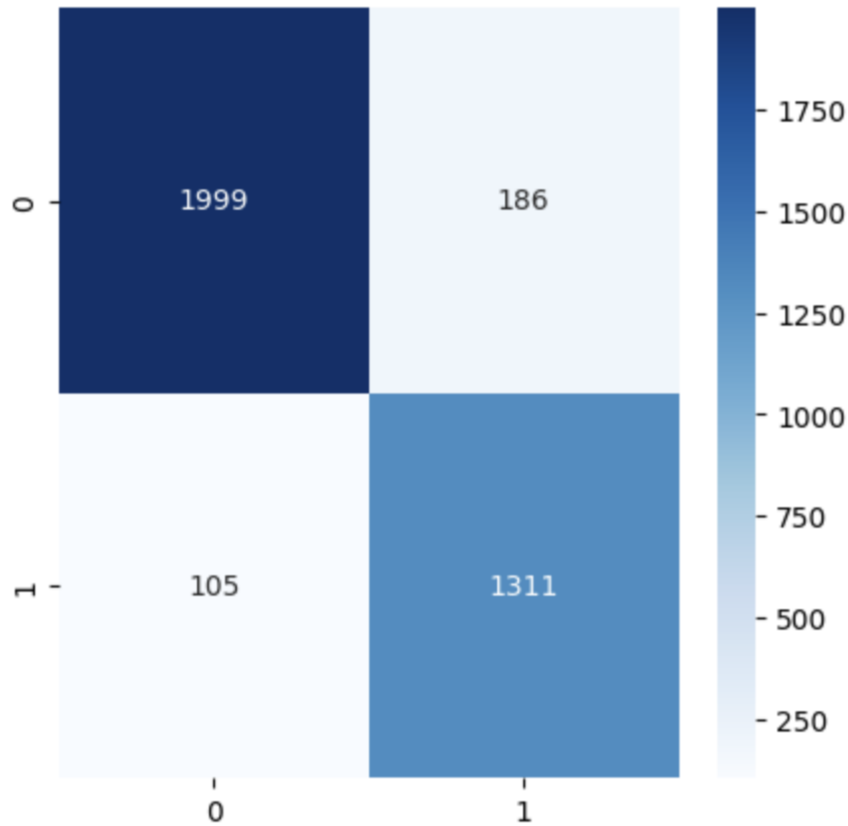
"ham precision"	0.95
"ham precision"	0.88
Accuracy	0.9191891141349625

The AdaBoost Ensemble Outputs the following: Here we need to remember that 0 represents "ham", and 1 represents "spam".

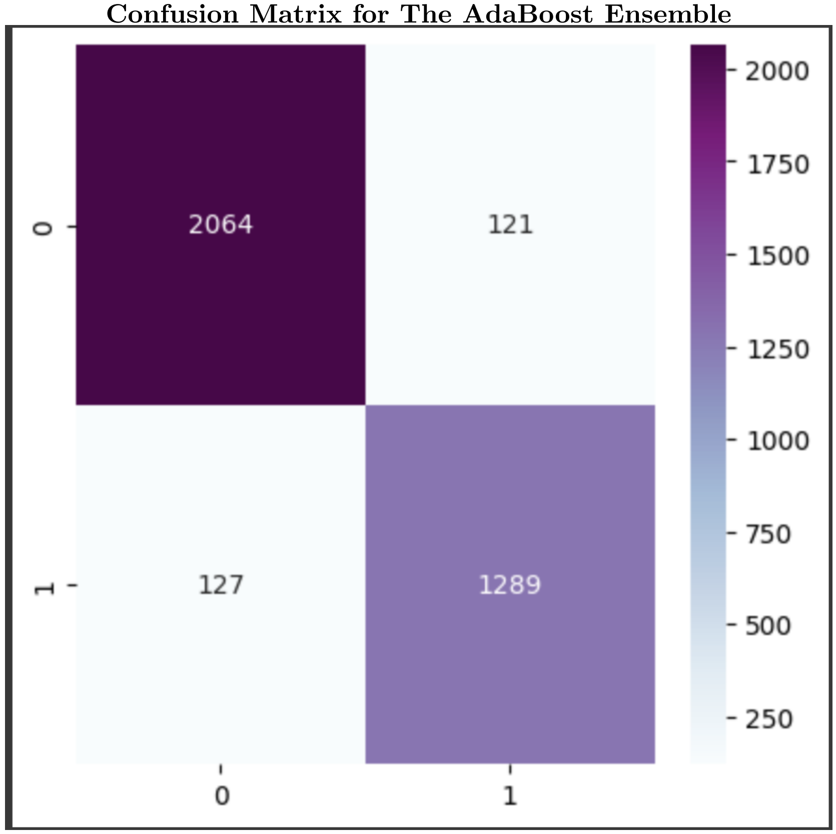
```
Prediction for 20 observation:     [0 0 1 0 1 1 1 0 1 1 0 0 0 0 1 1 0 0 0 1]
Actual values for 20 observation:  [0 0 1 0 0 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1]
[[2064  121]
 [ 127 1289]]
              precision    recall  f1-score   support

           0       0.94      0.94      0.94      2185
           1       0.91      0.91      0.91      1416

    accuracy                           0.93      3601
   macro avg       0.93      0.93      0.93      3601
weighted avg       0.93      0.93      0.93      3601

0.9311302415995557
```

"ham precision"	0.94
"ham precision"	0.91
Accuracy	0.9311302415995557

Confusion Matrix for The Fused Model



Confusion Matrix for The AdaBoost Ensemble



4 Second Task

4.1 Compare the accuracies of the fused model with Random Forest

Outputting 20 predicted values for the Fused Model

```
Prediction for 20 observation:     [0 0 1 0 1 1 1 0 1 1 0 0 0 0 1 1 0 1 1 1]
Actual values for 20 observation:  [0 0 1 0 0 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1]
```

Outputting 20 predicted values for the Random Forest Classifier

```
Prediction for 20 observation:     [0 0 1 0 0 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1]
Actual values for 20 observation:  [0 0 1 0 0 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1]
```

The Fused Model Outputs the following: Here we need to remember that 0 represents ”ham”, and 1 represents ”spam”

```
[[1999  186]
 [ 105 1311]]
              precision    recall  f1-score   support

           0       0.95      0.91      0.93      2185
           1       0.88      0.93      0.90      1416

    accuracy                           0.92      3601
   macro avg       0.91      0.92      0.92      3601
weighted avg       0.92      0.92      0.92      3601

0.9191891141349625
```

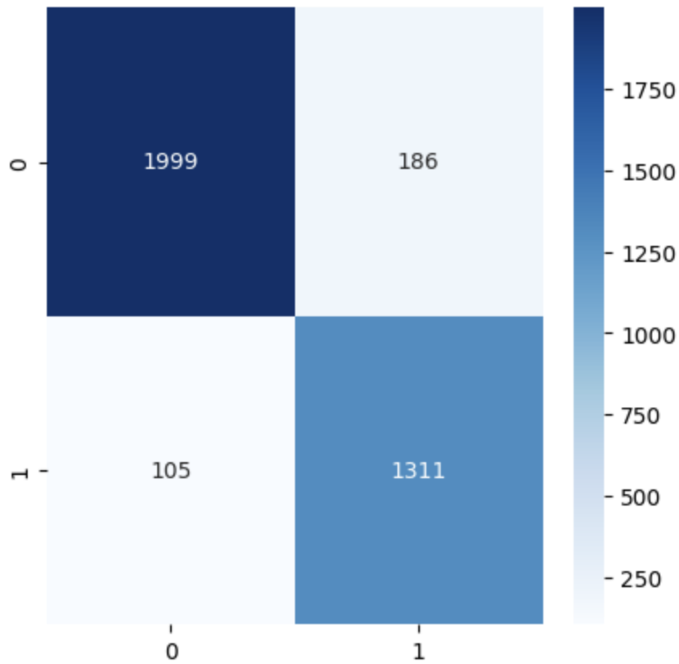
”ham precision”	0.95
”ham precision”	0.88
Accuracy	0.9191891141349625

The Random Forest Classifier Outputs the following: Here we need to remember that 0 represents "ham", and 1 represents "spam"

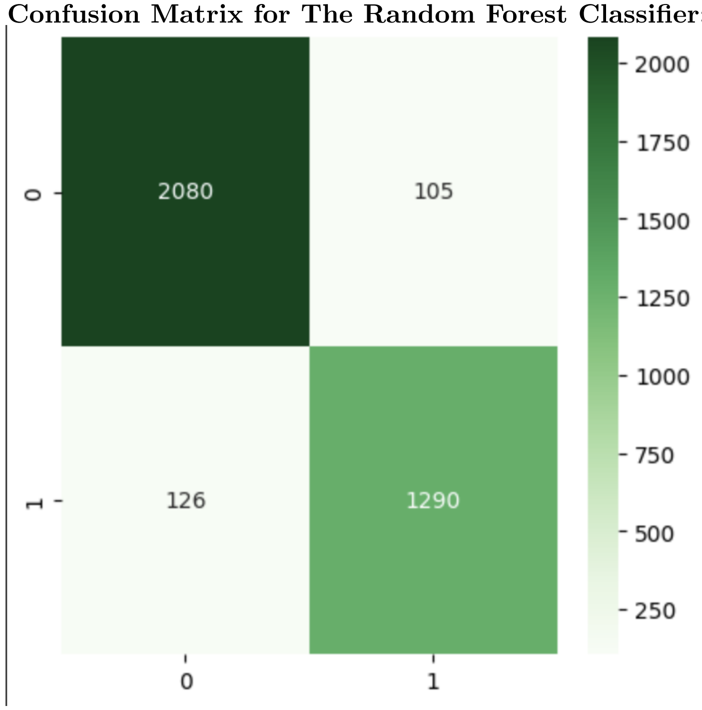
```
[[2080  105]
 [ 126 1290]]
              precision    recall  f1-score   support

           0       0.94      0.95      0.95      2185
           1       0.92      0.91      0.92      1416

    accuracy                           0.94      3601
   macro avg       0.93      0.93      0.93      3601
weighted avg       0.94      0.94      0.94      3601

0.9358511524576506
```

"ham precision"	0.94
"ham precision"	0.92
Accuracy	0.9358511524576506

Confusion Matrix for The Fused Model:



Confusion Matrix for The Random Forest Classifier:



5 The impact of training sample size on the accuracies of the fused classifier and the AdaBoost Ensemble with Decision Tree as the base learner.

5.1 Training-test splits: 50%-50%

```
#Create a training and test set
spam_training_set, spam_test_set = train_test_split(spam_dataset_dataframe, test_size = 0.5
, random_state = 42)
spam_dataset_dataframe.keys()
```

Outputting 20 predicted values for the Fused Model

```
Prediction for 20 observation:     [0 0 1 1 1 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1]
Actual values for 20 observation:  [0 0 1 0 0 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1]
```

Outputting 20 predicted values for AdaBoost Ensemble

```
Prediction for 20 observation:     [0 0 1 0 1 1 1 0 1 1 0 0 0 0 1 1 0 0 0 1]
Actual values for 20 observation:  [0 0 1 0 0 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1]
```

The Fused Model Outputs the following: Here we need to remember that 0 represents "ham", and 1 represents "spam"

```
[[1291  110]
 [  50  850]]
              precision    recall  f1-score   support

           0       0.96      0.92      0.94      1401
           1       0.89      0.94      0.91       900

    accuracy                           0.93      2301
   macro avg       0.92      0.93      0.93      2301
weighted avg       0.93      0.93      0.93      2301

0.930465015210778
```

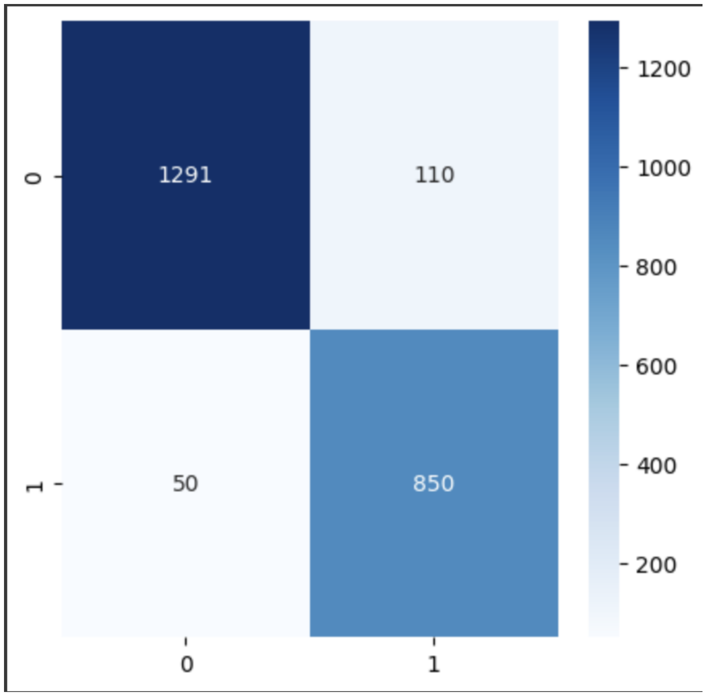
"ham precision"	0.96
"ham precision"	0.89
Accuracy	0.930465015210778

The AdaBoost Ensemble Outputs the following: Here we need to remember that 0 represents "ham", and 1 represents "spam"

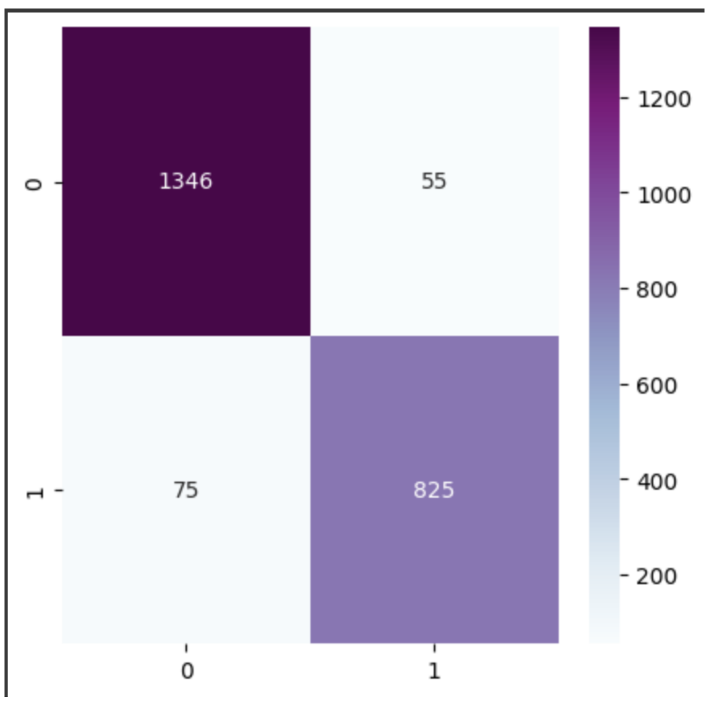
```
[[1346   55]
 [  75  825]]
              precision    recall  f1-score   support

           0       0.95      0.96      0.95      1401
           1       0.94      0.92      0.93       900

    accuracy                           0.94      2301
   macro avg       0.94      0.94      0.94      2301
weighted avg       0.94      0.94      0.94      2301

0.943502824858757
```

"ham precision"	0.95
"ham precision"	0.94
Accuracy	0.943502824858757

Confusion Matrix for The Fused Model:



Confusion Matrix for The AdaBoost Ensemble



5.2 Training-test splits: 60%-40%

```
#Create a training and test set
spam_training_set, spam_test_set = train_test_split(spam_dataset_dataframe, test_size = 0.4
, random_state = 42)
spam_dataset_dataframe.keys()
```

Outputting 20 predicted values for the Fused Model

```
Prediction for 20 observation:     [0 0 1 0 1 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1]
Actual values for 20 observation:  [0 0 1 0 0 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1]
```

Outputting 20 predicted values for AdaBoost Ensemble

```
Prediction for 20 observation:     [0 0 1 0 1 1 1 0 1 1 0 0 0 0 1 1 0 0 0 1]
Actual values for 20 observation:  [0 0 1 0 0 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1]
```

The Fused Model Outputs the following: Here we need to remember that 0 represents "ham", and 1 represents "spam"

```
[[1048   86]
 [  46  661]]
              precision    recall  f1-score   support

           0       0.96      0.92      0.94      1134
           1       0.88      0.93      0.91       707

    accuracy                           0.93      1841
   macro avg       0.92      0.93      0.92      1841
weighted avg       0.93      0.93      0.93      1841

0.9282998370450842
```

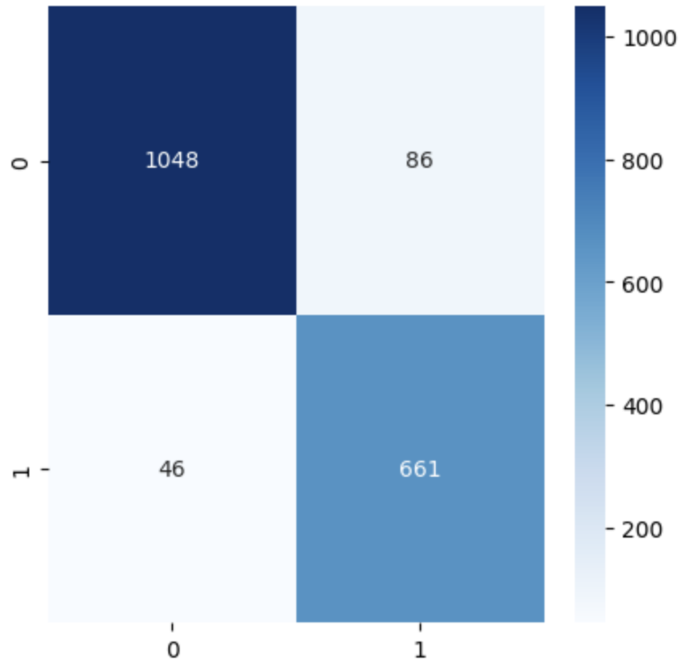
"ham precision"	0.96
"ham precision"	0.88
Accuracy	0.9282998370450842

The AdaBoost Ensemble Outputs the following: Here we need to remember that 0 represents "ham", and 1 represents "spam"

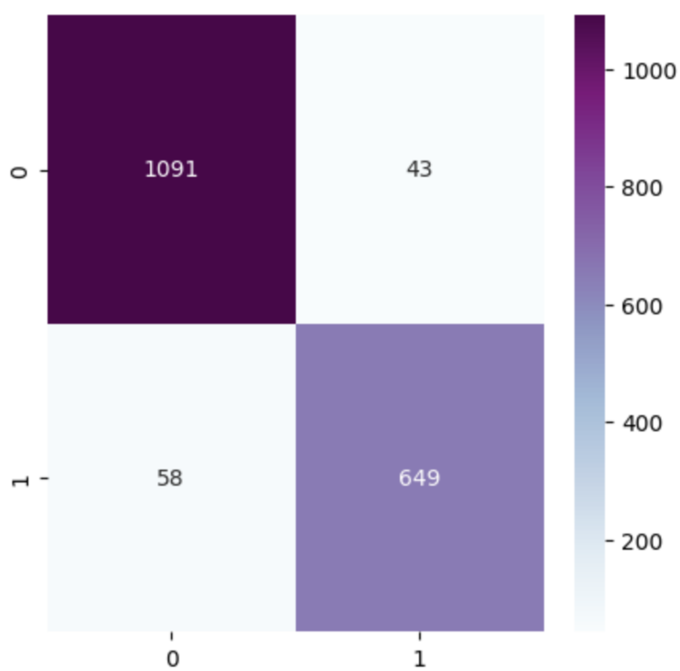
```
[[1091   43]
 [  58  649]]
              precision    recall  f1-score   support

           0       0.95      0.96      0.96      1134
           1       0.94      0.92      0.93       707

    accuracy                           0.95      1841
   macro avg       0.94      0.94      0.94      1841
weighted avg       0.95      0.95      0.95      1841

0.9451385116784357
```

"ham precision"	0.95
"ham precision"	0.94
Accuracy	0.9451385116784357

Confusion Matrix for The Fused Model:



Confusion Matrix for The AdaBoost Ensemble



5.3 Training-test splits: 70%-30%

```
#Create a training and test set
spam_training_set, spam_test_set = train_test_split(spam_dataset_dataframe, test_size = 0.3
, random_state = 42)
spam_dataset_dataframe.keys()
```

Outputting 20 predicted values for the Fused Model

```
Prediction for 20 observation:     [0 0 1 0 0 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1]
Actual values for 20 observation:  [0 0 1 0 0 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1]
```

Outputting 20 predicted values for AdaBoost Ensemble

```
Prediction for 20 observation:     [0 0 1 0 1 1 1 0 1 1 0 0 0 0 1 1 0 0 0 1]
Actual values for 20 observation:  [0 0 1 0 0 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1]
```

The Fused Model Outputs the following: Here we need to remember that 0 represents "ham", and 1 represents "spam"

```
[[776  61]
 [ 33 511]]
              precision    recall  f1-score   support

           0       0.96      0.93      0.94       837
           1       0.89      0.94      0.92       544

    accuracy                           0.93      1381
   macro avg       0.93      0.93      0.93      1381
weighted avg       0.93      0.93      0.93      1381

0.9319333816075308
```

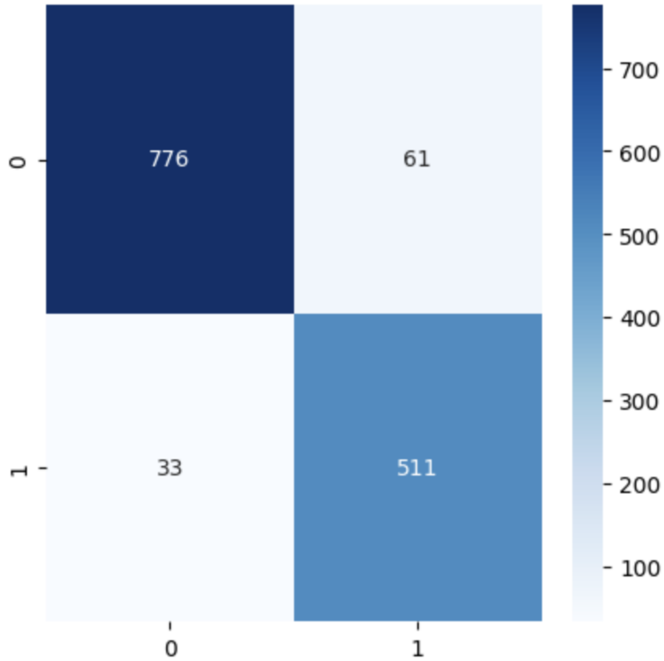
"ham precision"	0.96
"ham precision"	0.89
Accuracy	0.9319333816075308

The AdaBoost Ensemble Outputs the following: Here we need to remember that 0 represents ”ham”, and 1 represents ”spam”

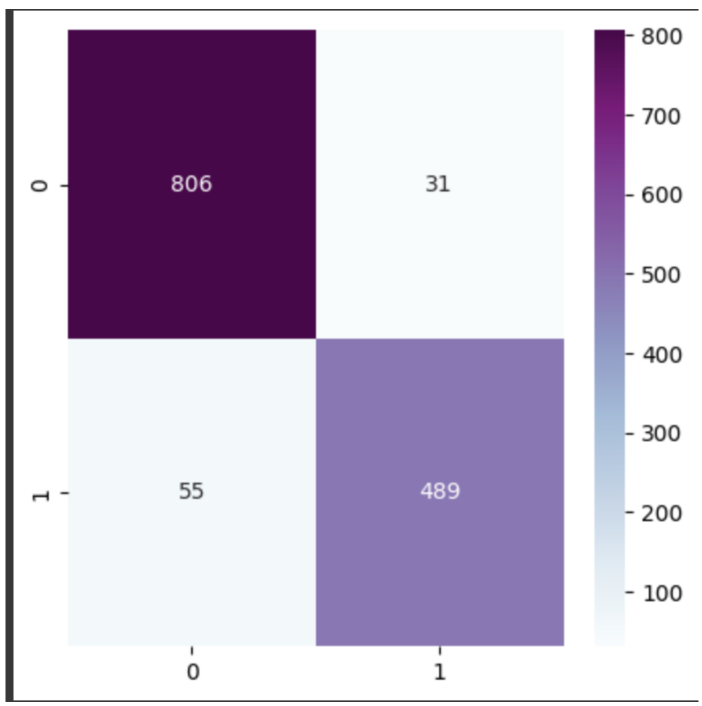
```
[[806  31]
 [ 55 489]]
              precision    recall  f1-score   support

           0       0.94      0.96      0.95       837
           1       0.94      0.90      0.92       544

    accuracy                           0.94      1381
   macro avg       0.94      0.93      0.93      1381
weighted avg       0.94      0.94      0.94      1381

0.9377262853005068
```

”ham precision”	0.94
”ham precision”	0.94
Accuracy	0.9377262853005068

Confusion Matrix for The Fused Model:



Confusion Matrix for The AdaBoost Ensemble



5.4 Training-test splits: 80%-20%

```
#Create a training and test set
spam_training_set, spam_test_set = train_test_split(spam_dataset_dataframe, test_size = 0.2
, random_state = 42)
spam_dataset_dataframe.keys()
```

Outputting 20 predicted values for the Fused Model

```
Prediction for 20 observation:     [0 0 1 0 0 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1]
Actual values for 20 observation:  [0 0 1 0 0 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1]
```

Outputting 20 predicted values for AdaBoost Ensemble

```
Prediction for 20 observation:     [0 0 1 0 1 1 1 0 1 1 0 0 0 0 1 1 0 0 0 1]
Actual values for 20 observation:  [0 0 1 0 0 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1]
```

The Fused Model Outputs the following: Here we need to remember that 0 represents ”ham”, and 1 represents ”spam”

```
[[513  42]
 [ 24 342]]
              precision    recall  f1-score   support

           0       0.96      0.92      0.94       555
           1       0.89      0.93      0.91       366

    accuracy                           0.93       921
   macro avg       0.92      0.93      0.93       921
weighted avg       0.93      0.93      0.93       921

0.9283387622149837
```

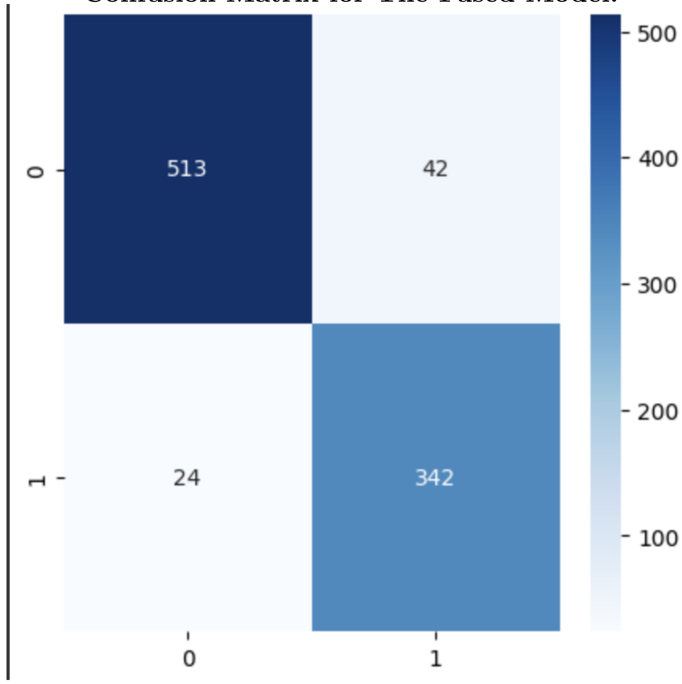
”ham precision”	0.96
”ham precision”	0.89
Accuracy	0.9283387622149837

The AdaBoost Ensemble Outputs the following: Here we need to remember that 0 represents ”ham”, and 1 represents ”spam”

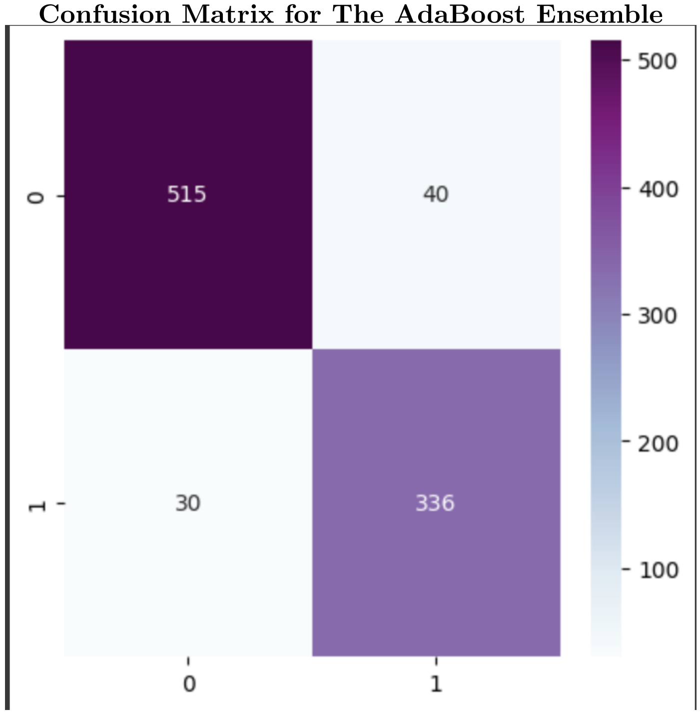
```
[[515  40]
 [ 30 336]]
              precision    recall  f1-score   support

           0       0.94      0.93      0.94       555
           1       0.89      0.92      0.91       366

    accuracy                           0.92       921
   macro avg       0.92      0.92      0.92       921
weighted avg       0.92      0.92      0.92       921

0.9239956568946797
```

”ham precision”	0.94
”ham precision”	0.89
Accuracy	0.9239956568946797

Confusion Matrix for The Fused Model:



Confusion Matrix for The AdaBoost Ensemble



Conclusion

From the above observation, we can reach the conclusion that:

- AdaBoost Ensemble has higher accuracy than the Fusion Model.
- Random Forest Classifier has higher precision than Fusion Model.
- Comparing AdaBoost Ensemble and the Fusion Model accuracies with the following training-test splits: 50%-50%, 60%-40%, 70%-30%, and 80%-20%, we noticed that the fused model has higher accuracy than AdaBoost Ensemble. The accuracy of the fused model reaches its peak at the division 60%-40% and then starts decreasing, meanwhile, the accuracy of the AdaBoost Ensemble reaches its peak on the 70%-30% division but never passes the accuracy of the fused model.