Data Mining

Assignment 3

Data Classification

Aya Lotfy Saeed 4 **Dahlia Chehata 27**

Table of contents

Data description

This dataset is generated to simulate registration of high energy gamma particles in a ground-based atmospheric Cherenkov gamma telescope using the imaging technique.
The dataset consists of 2 classes:gammas (signal) and hadrons (background).
There are 12332 tuples with class g and 6688 tuples with class h.

Balancing data

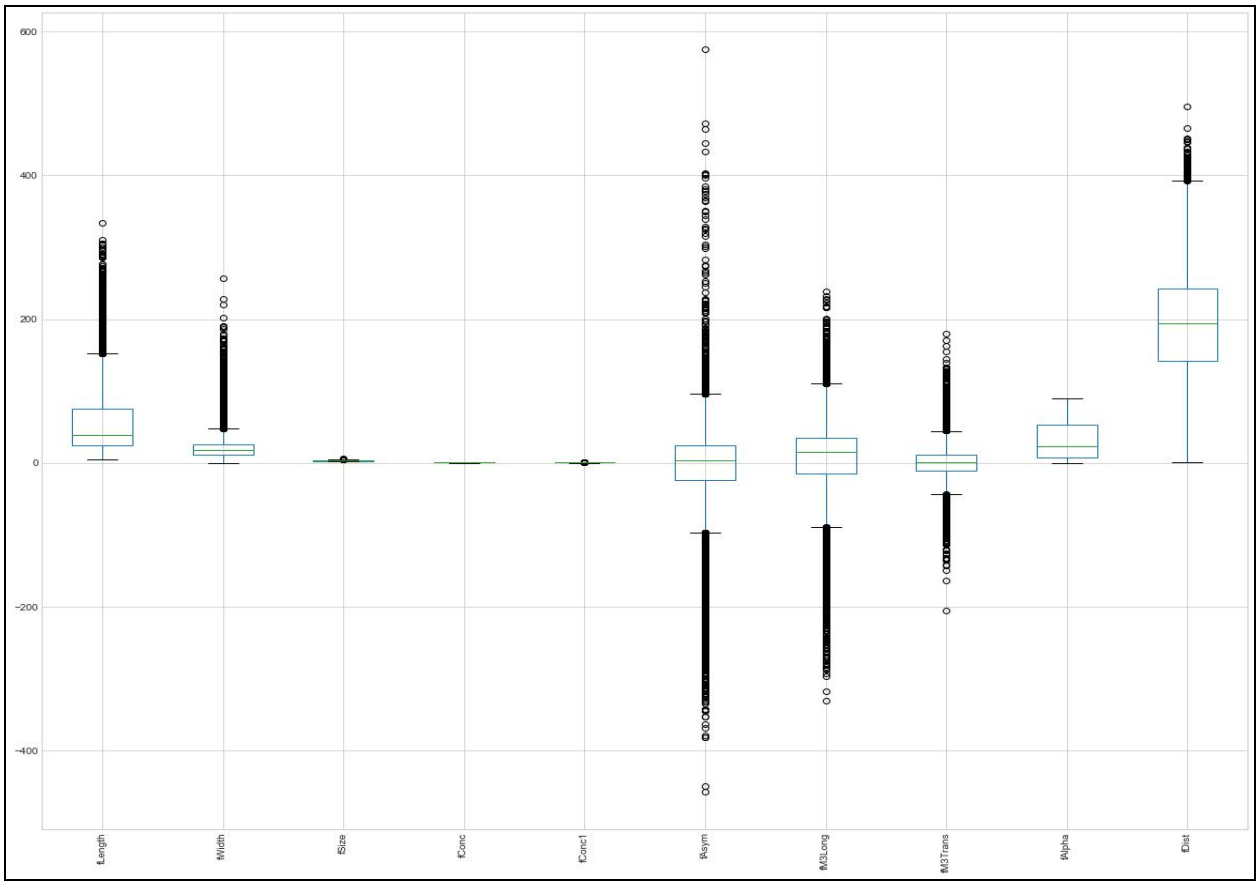
we used downsampling majority approach to equalize both classes each wit 6688 tuples

- **description after balancing**

Index	fLength	fWidth	fSize	fConc	fConc1	fAsym	fM3Long	fM3Trans	fAlpha	fDist
count	13376	13376	13376	13376	13376	13376	13376	13376	13376	13376
mean	57.2106	23.6668	2.84157	0.379196	0.214591	-7.60507	7.56761	0.268904	31.3483	195.218
std	46.8919	20.8917	0.475124	0.18484	0.112075	65.5175	56.308	23.2285	26.9191	76.5234
min	4.2835	0	1.9413	0.0131	0.0003	-457.916	-331.78	-205.895	0	1.2826
25%	24.7425	11.702	2.4955	0.2326	0.1265	-24.5678	-14.8528	-11.0041	7.32503	142.25
50%	38.6189	17.1085	2.7547	0.35575	0.1972	2.75945	14.717	0.4203	23.422	193.754
75%	75.6808	26.0966	3.11765	0.505925	0.2872	23.7607	35.0899	11.1372	52.3546	242.618
max	334.177	256.382	5.3233	0.893	0.643	575.241	238.321	179.851	90	495.561

Visualisation

- **Box Plots**



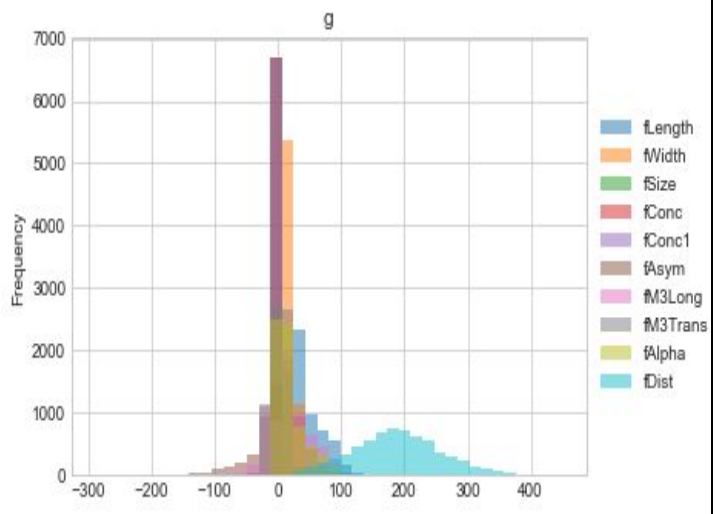
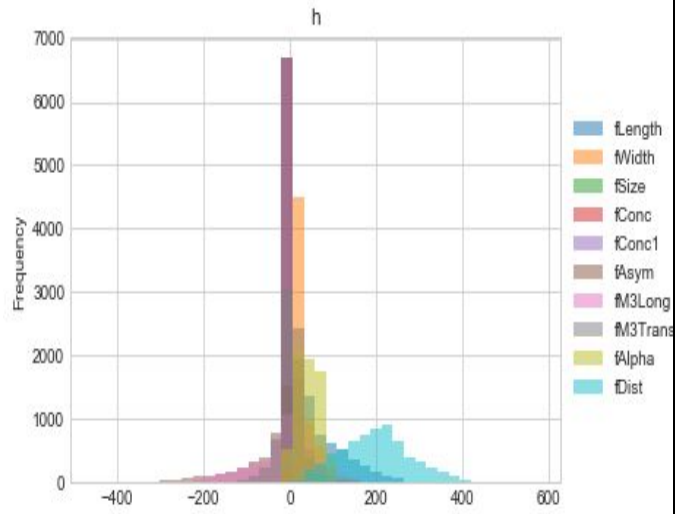
Observations:

- The dataset attributes are different in ranges
- **fAsym** and **fM3Long** attributes have huge outliers (data need to be normalized)
- **fConc** has a zero standard deviation because it has a nearly fixed value (redundant dimension to be removed).
- **fAlpha** have a normal distribution and no outliers

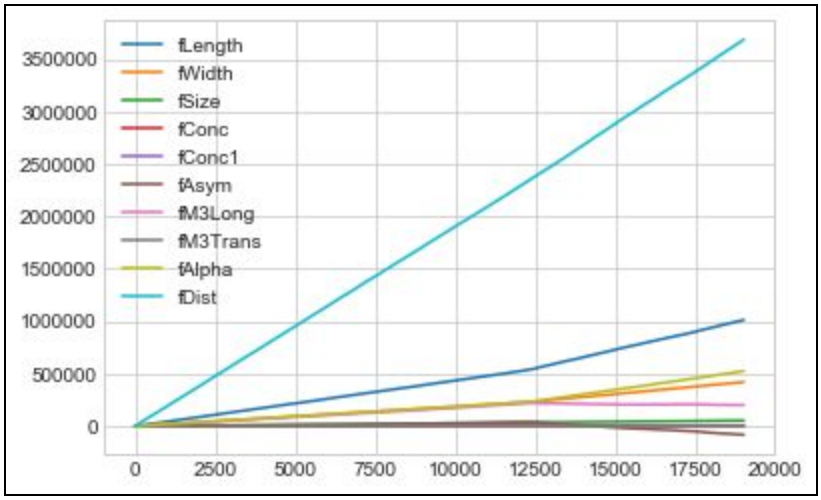
Conclusion

- The data needs to be normalized
- Outliers need to be removed
- Redundant attributes need to be removed

- **Histograms**

	
we notice the fAsym attribute has a very large frequency as well as fWidth	we notice the fAsym attribute has a very large frequency as well as fWidth

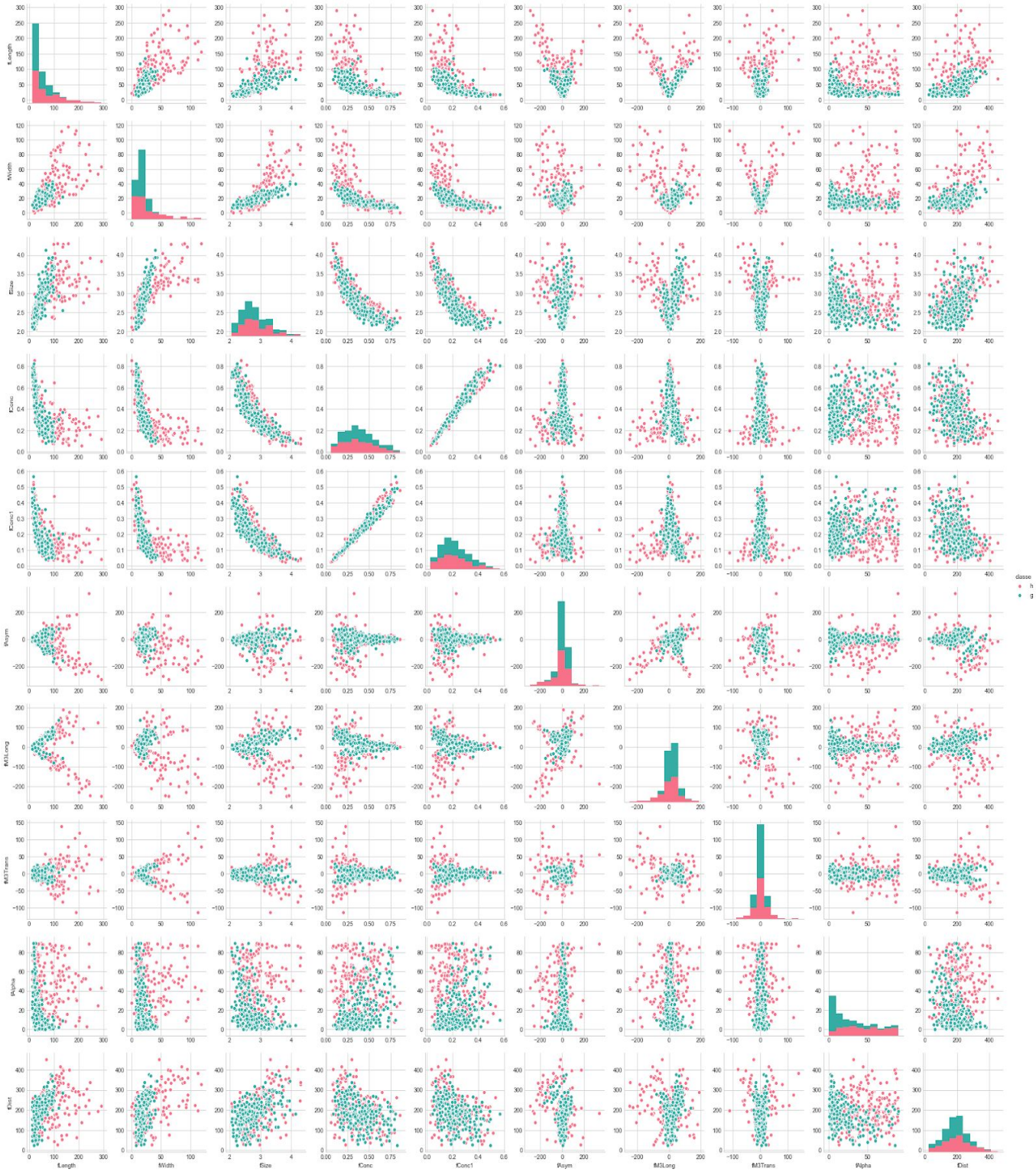
- **Line plots**



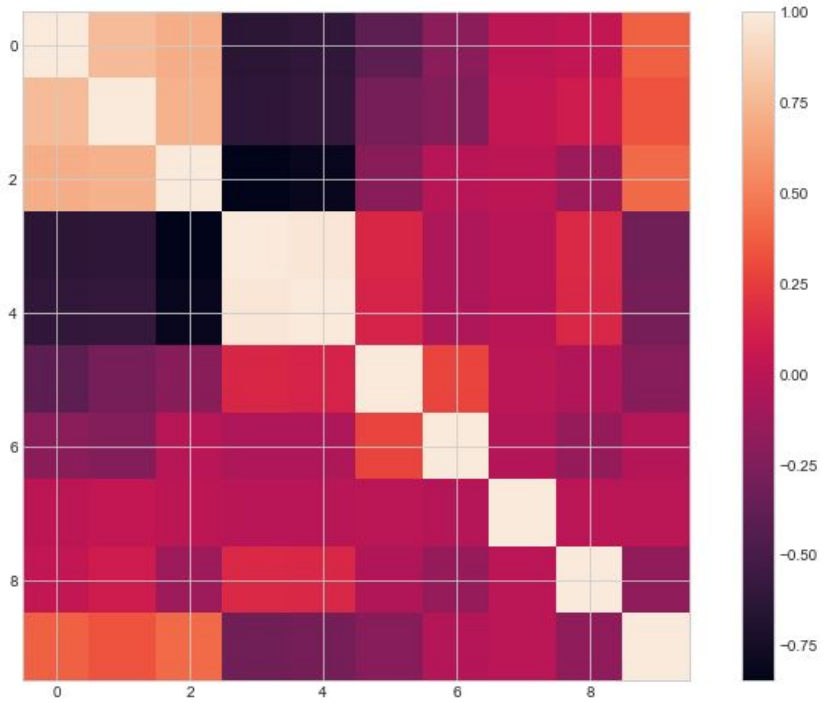
We notice the range of values for each attribute where fDist is linearly increasing with the tuple index

- **Scatter plots**

taking only 500 samples to avoid graph density

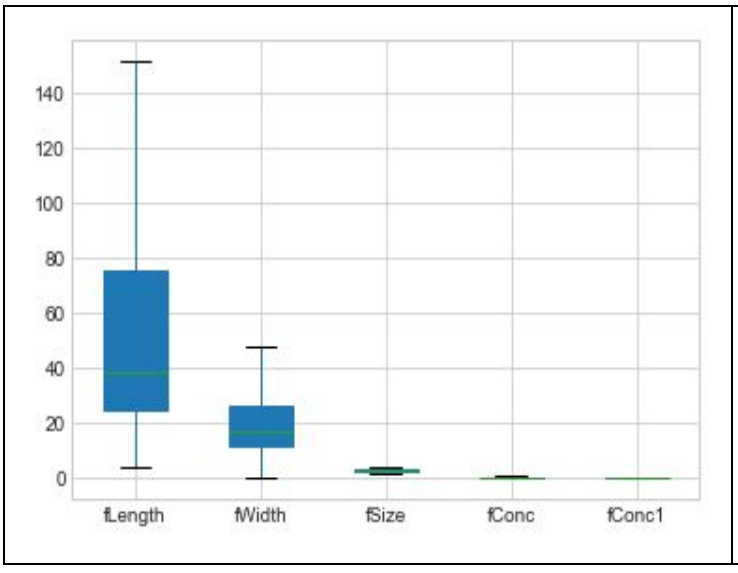


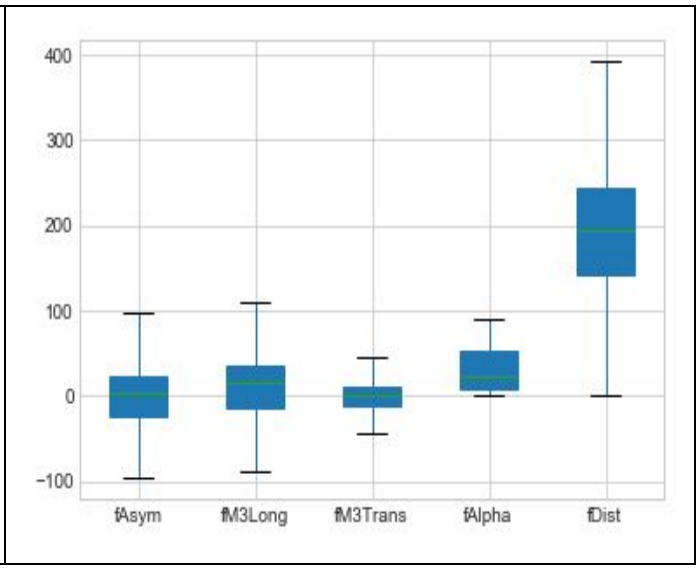
- **Pearson Correlation matrix**



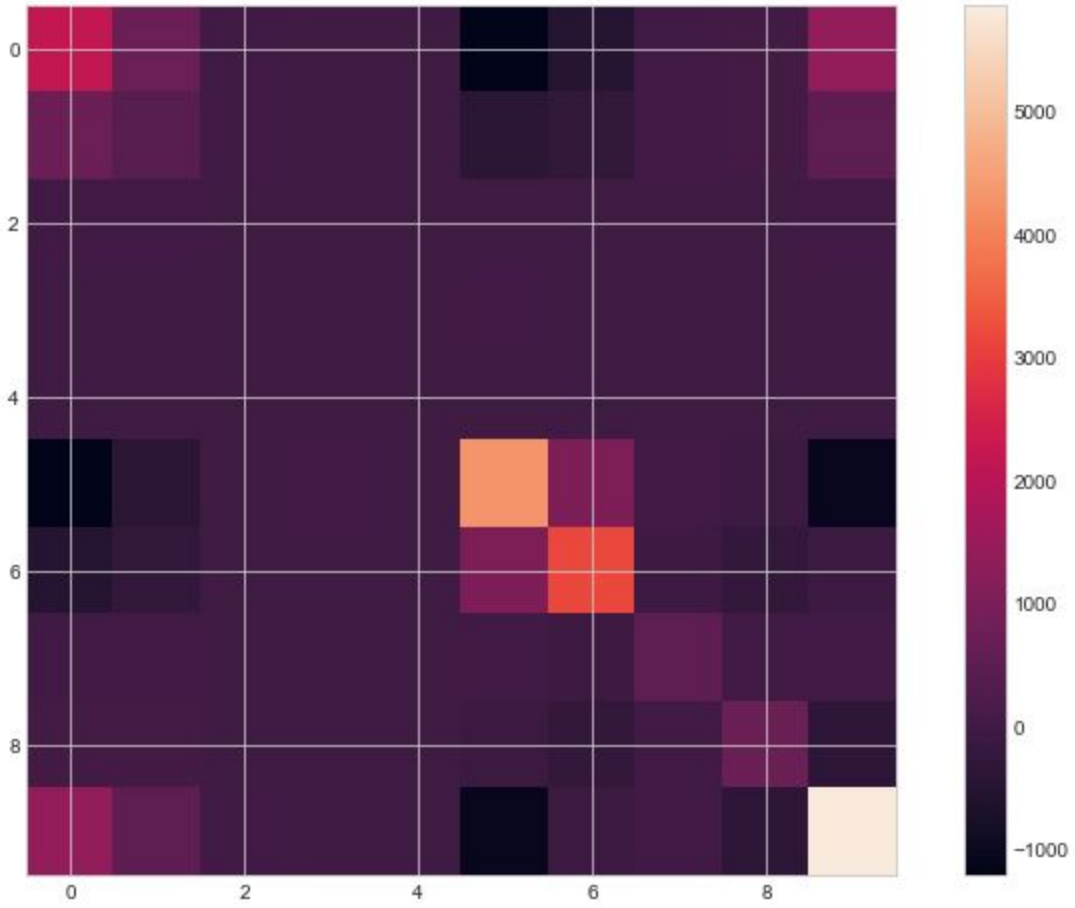
Observation

- By using the correlation matrix, that there is high correlation among fLength, fWidth, and fSize features. Another slight correlation between fDist feature and the first 3 features.
- Box plots can assert our previous claim of features have different ranges of values which may affect the learning process.





- **Covariance matrix**



Data split

We split the dataset to 70 % training set and 30 % testing set

- Training set has 9363 samples.
- Testing set has 4013 samples.

Preprocessing

involves 2 main steps:

Z normalisation:

According to the previous observations, data have to be normalized. We use z-normalization to normalize the data. We are not going use min-max since there are a lot of outliers in each feature..

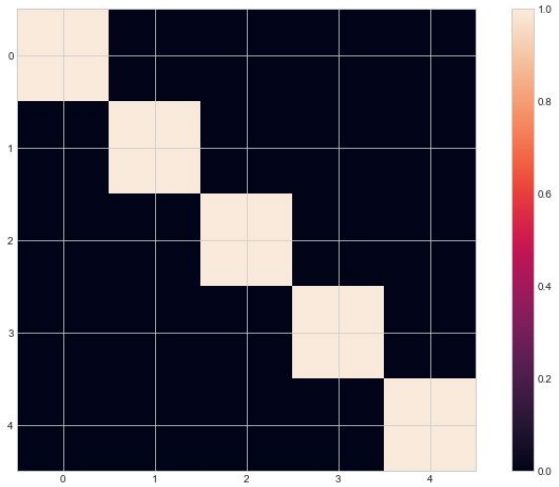
PCA feature selection/extraction

We also apply PCA to reduce the correlated, unnecessary features in the dataset first we examine different number of features and compute the variance sum to determine the best number of components:

	captured variance sum	component number
0	0.423712	1
1	0.574513	2
2	0.676609	3
3	0.776215	4
4	0.851559	5
5	0.918191	6
6	0.960291	7
7	0.982126	8
8	0.997878	9
9	1.000000	10

We can notice than only 5 components are relevant and the rest are redundant (highly correlated)

Data correlation matrix after processing:



The attributes are uncorrelated after processing and outliers are removed

After preprocessing :
data becomes:

	p0	p1	p2	p3	p4	classe
0	1.353216	-1.602735	-0.545332	0.768234	0.061246	g
1	-1.351668	-0.267834	-0.600451	0.034977	-0.273350	g
2	1.112049	-1.276764	0.544860	-0.437516	0.742278	g
3	-2.599330	0.843268	-0.850632	0.008230	0.965704	g
4	0.634342	-0.479289	-0.662526	0.281333	-1.178775	g

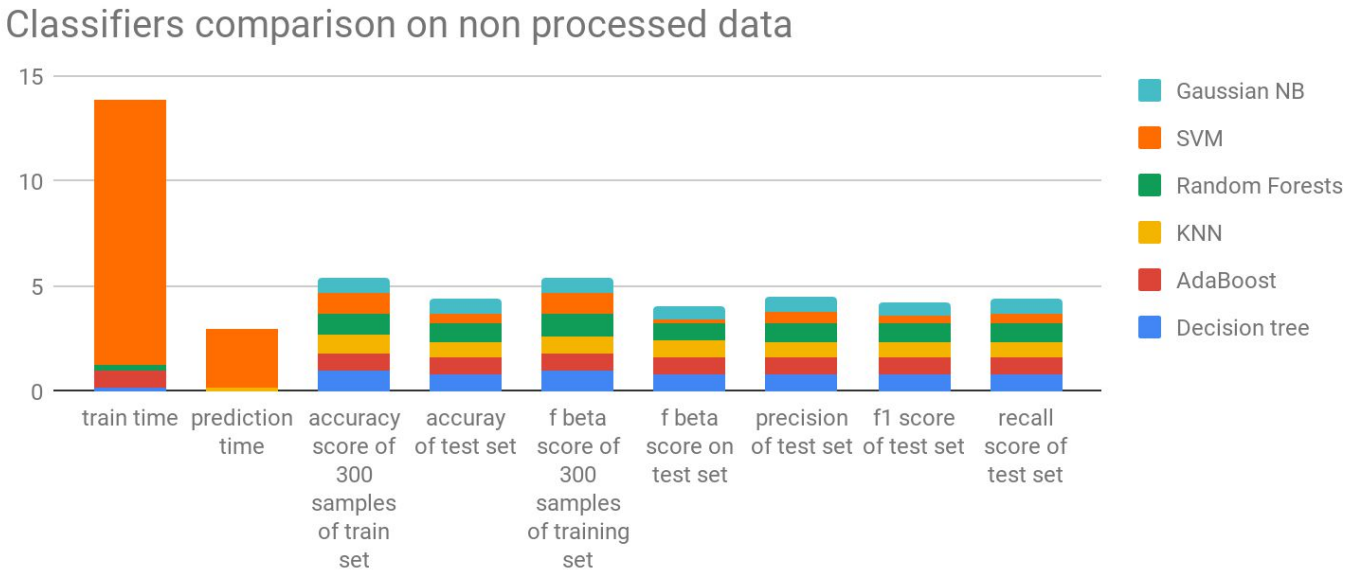
data information:

Index	p0	p1	p2	p3	p4
count	13376	13376	13376	13376	13376
mean	-9.09726e-16	9.84958e-16	-2.97343e-16	-6.86253e-17	4.15138e-1
std	2.0585	1.22806	1.01046	0.998066	0.86804
min	-4.3347	-3.44646	-3.63014	-6.78001	-3.80388
25%	-1.50068	-0.800189	-0.633021	-0.534138	-0.586064
50%	-0.281122	-0.0777022	-0.06317	-0.0169598	-0.0274029
75%	1.25709	0.665024	0.57874	0.464234	0.562816
max	8.98456	6.75501	7.40646	9.83514	4.12969

Classification

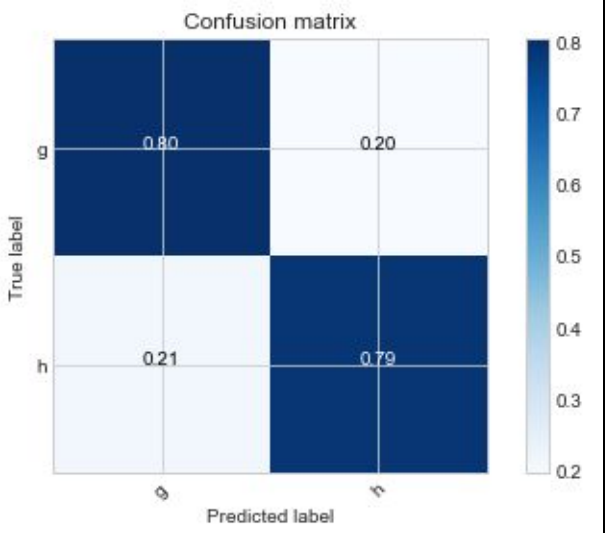
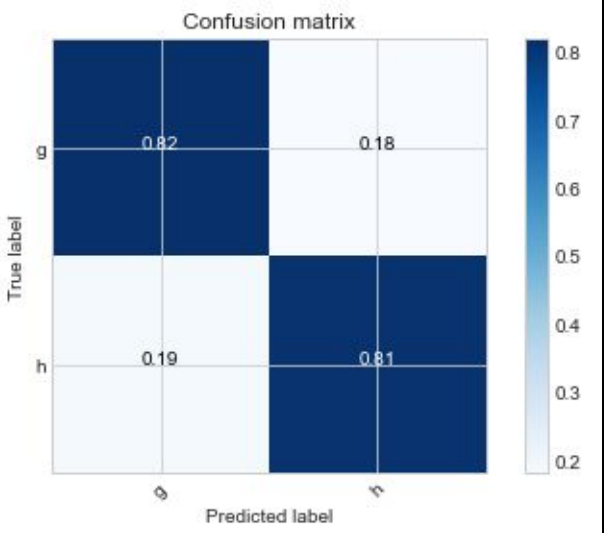
- before preprocessing results of all models

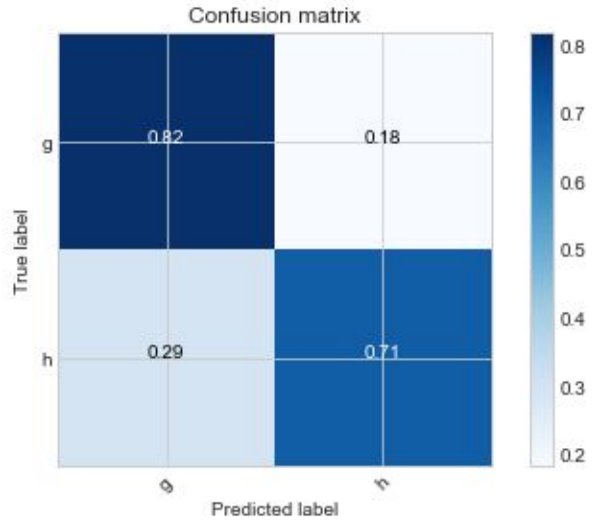
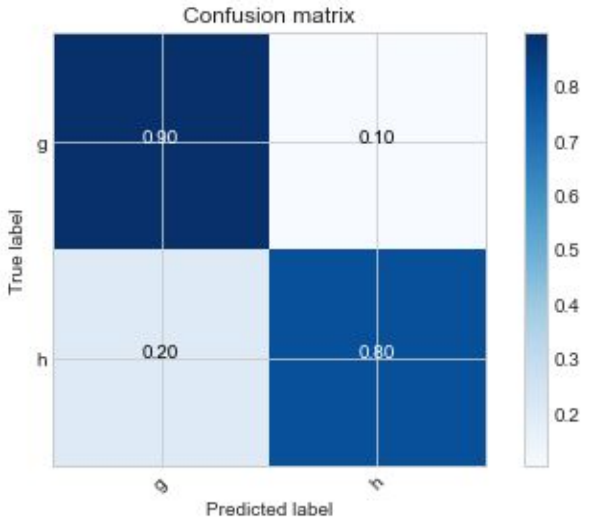
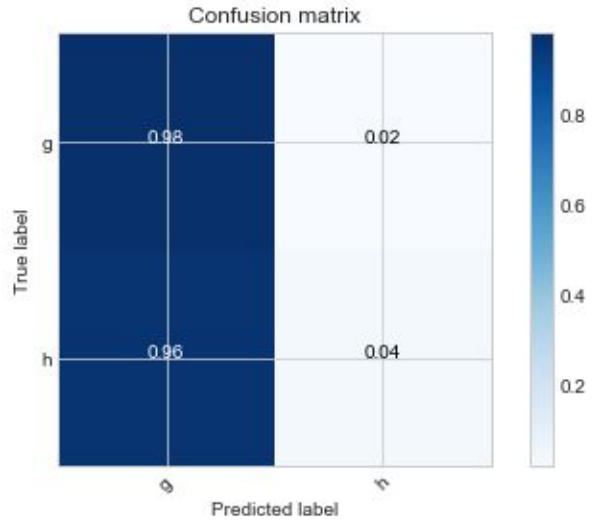
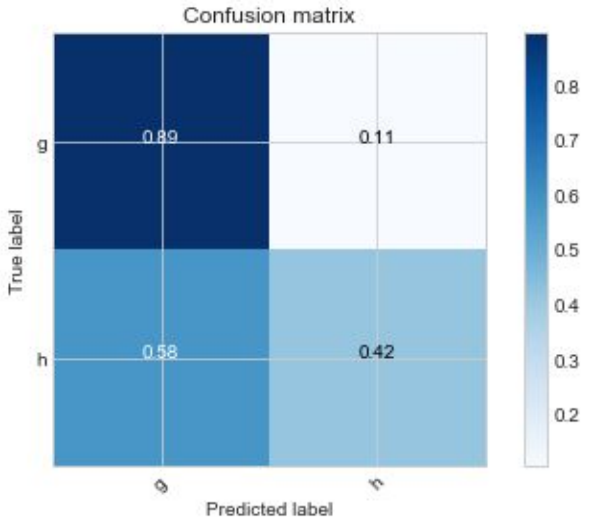
```
DecisionTreeClassifier
DecisionTreeClassifier trained on 9363 samples.
AdaBoostClassifier
AdaBoostClassifier trained on 9363 samples.
KNeighborsClassifier
KNeighborsClassifier trained on 9363 samples.
RandomForestClassifier
RandomForestClassifier trained on 9363 samples.
SVC
SVC trained on 9363 samples.
GaussianNB
GaussianNB trained on 9363 samples.
```



- ## Confusion matrices before preprocessing

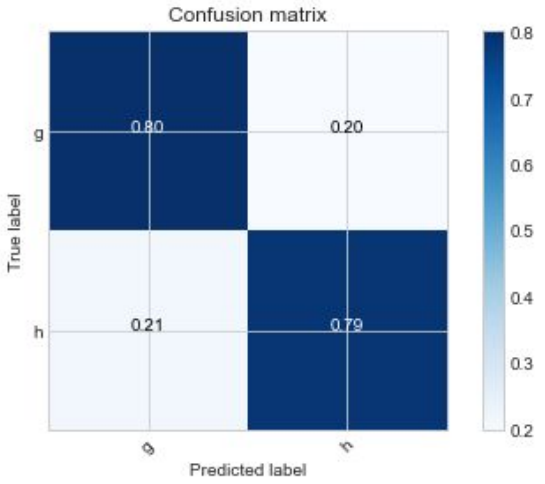
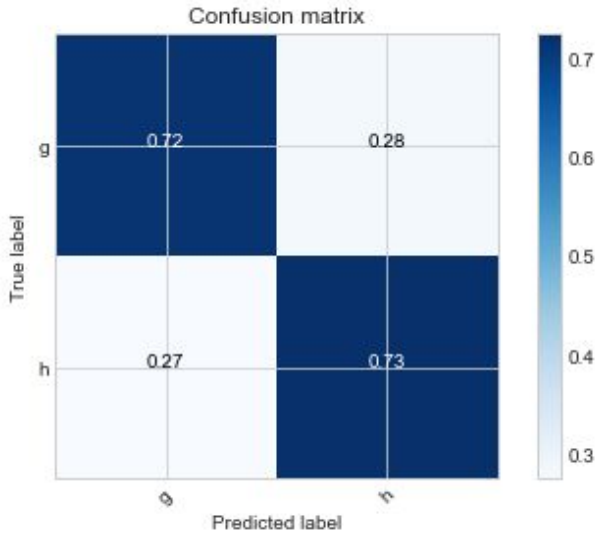
By definition a confusion matrix C is such that Cij is equal to the number of observations known to be in group i but predicted to be in group j.
Thus in binary classification, the count of **true negatives** is c00 , **false negatives** is c10, **true positives** is c11 and **false positives** is c01.

Decision Tree	AdaBoost
	

KNN	Random Forests
	
SVM	**Gaussian**
	

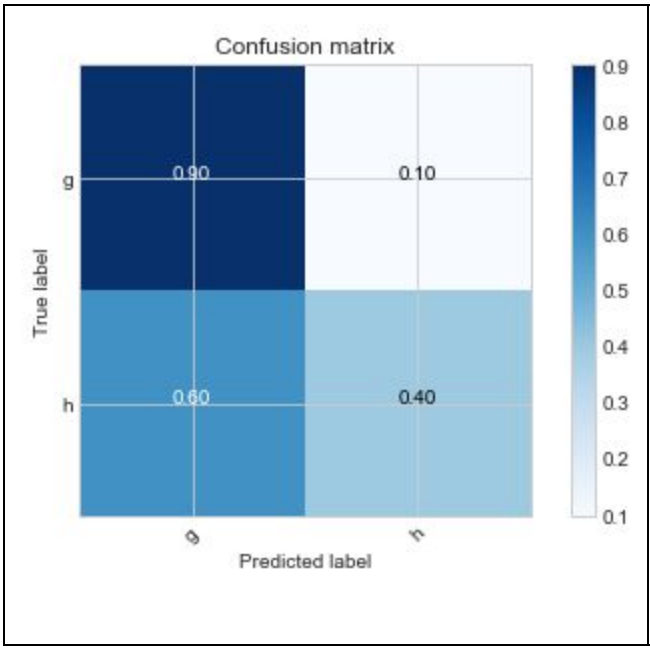
Classifiers comparison before and after preprocessing

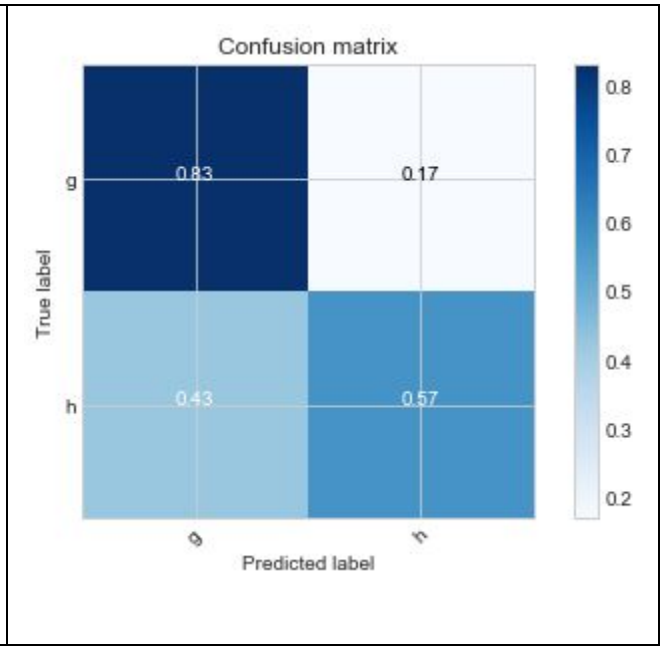
- **Decision Tree**

before preprocessing	After preprocessing
testing accuracy is : 0.7946673311736855 precision_score is : 0.7947370406423341 recall_score is : 0.7946673311736855 F Measure is : 0.7946577680923154 Normalized confusion matrix [[0.80189621 0.19810379] [0.21254355 0.78745645]] 	testing accuracy is : 0.7211562422128084 precision_score is : 0.7211504486367339 recall_score is : 0.7211562422128084 F Measure is : 0.7211511510438317 Normalized confusion matrix [[0.71694319 0.28305681] [0.27470356 0.72529644]] 

- **Gaussian Naive Bayesian**

before preprocessing	After preprocessing
testing accuracy is : 0.6496386743084974 precision_score is : 0.7014434725165661 recall_score is : 0.6496386743084974 F Measure is : 0.6257770114177219 Normalized confusion matrix [[0.90269461 0.09730539] [0.60278746 0.39721254]]	testing accuracy is : 0.699975080986793 precision_score is : 0.7153939256986601 recall_score is : 0.699975080986793 F Measure is : 0.6950903929781364 Normalized confusion matrix [[0.82956259 0.17043741] [0.42737154 0.57262846]]



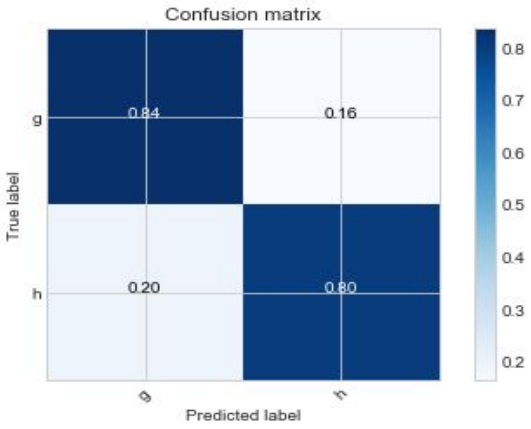


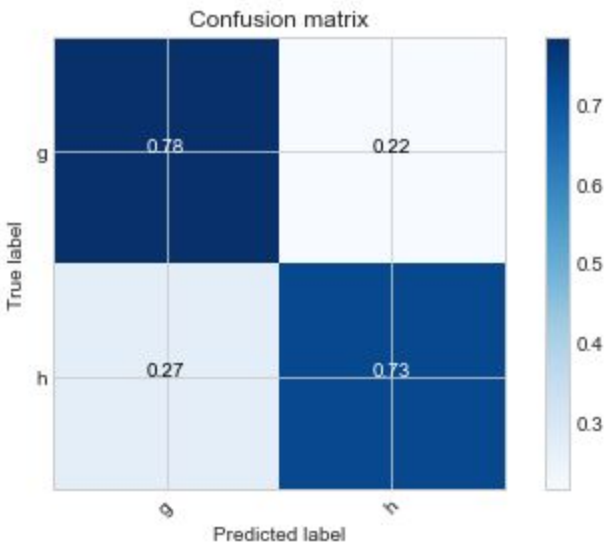
- **AdaBoost:**

  We can see from the following results that the AdaBoostClassifier model achieves a maximum accuracy result of 82% at number of estimators = 1000, using default base estimator as DecisionTreeClassifier(max_depth=1) on . It can be noticed that the increase of the number of estimators results in a better accuracy, but takes a longer time to fit and predict.

before preprocessing	**After preprocessing**
• **tuning parameters:** n_estimators is equal : 10 Accuracy: 0.78 (+/- 0.02) training accuracy is : 0.7914130086510733 n_estimators is equal : 120 Accuracy: 0.82 (+/- 0.01) training accuracy is : 0.8389405105201324 n_estimators is equal : 230 Accuracy: 0.82 (+/- 0.01) training accuracy is : 0.8442806792694649 n_estimators is equal : 340 Accuracy: 0.82 (+/- 0.00)	• **tuning parameters:** n_estimators is equal : 10 Accuracy: 0.74 (+/- 0.05) training accuracy is : 0.7559542881555057 n_estimators is equal : 120 Accuracy: 0.76 (+/- 0.02) training accuracy is : 0.7757129125280359 n_estimators is equal : 230 Accuracy: 0.76 (+/- 0.02) training accuracy is : 0.779023817152622 n_estimators is equal : 340 Accuracy: 0.76 (+/- 0.02) training accuracy is : 0.7818007049022749

training accuracy is : 0.8480187973939977 n_estimators is equal : 450 Accuracy: 0.82 (+/- 0.01) training accuracy is : 0.8515433087685571 n_estimators is equal : 560 Accuracy: 0.82 (+/- 0.01) training accuracy is : 0.8549610167681299 n_estimators is equal : 670 Accuracy: 0.82 (+/- 0.01) training accuracy is : 0.8577379045177828 n_estimators is equal : 780 Accuracy: 0.82 (+/- 0.00) training accuracy is : 0.8592331517675958 n_estimators is equal : 890 Accuracy: 0.82 (+/- 0.01) training accuracy is : 0.8600875787674891 n_estimators is equal : 1000 Accuracy: 0.82 (+/- 0.01) training accuracy is : 0.8623304496422087	n_estimators is equal : 450 Accuracy: 0.76 (+/- 0.02) training accuracy is : 0.7828687386521415 n_estimators is equal : 560 Accuracy: 0.76 (+/- 0.02) training accuracy is : 0.7844707892769411 n_estimators is equal : 670 Accuracy: 0.76 (+/- 0.02) training accuracy is : 0.7863932500267008 n_estimators is equal : 780 Accuracy: 0.76 (+/- 0.02) training accuracy is : 0.7868204635266475 n_estimators is equal : 890 Accuracy: 0.76 (+/- 0.02) training accuracy is : 0.7882089074014739 n_estimators is equal : 1000 Accuracy: 0.76 (+/- 0.02) training accuracy is : 0.7895973512763004
n estimators =1000 with accuracy 86.2% on training set and 82 % on test set testing accuracy is : 0.8193371542486918 precision_score is : 0.8196798968086488 recall_score is : 0.8193371542486918 F Measure is : 0.819293392745702 **Normalized confusion matrix** [[0.83532934 0.16467066] [0.19661523 0.80338477]]	**n estimators =1000 with accuracy 82% on training set and 76% on test set** testing accuracy is : 0.7552952903065039 precision_score is : 0.7562742920810154 recall_score is : 0.7552952903065039 F Measure is : 0.755152458809285 **Normalized Confusion matrix** [[1556 433] [549 1475]]


Confusion matrix
g
h
0.84
0.16
0.20
0.80
True label
g
h
Predicted label
0.8
0.7
0.6
0.5
0.4
0.3
0.2


Confusion matrix
g
h
0.78
0.22
0.27
0.73
True label
g
h
Predicted label
0.7
0.6
0.5
0.4
0.3

- **KNN**
  We can see from the following results that the best accuracy of 76.7% is achieved at k = 17. which is less than the accuracy we achieved using AdaBoost (86.2%), but it is noticed that the time taken to fit the model and predict is way less than that of AdaBoost (0.17s vs 57s).

before preprocessing	**After preprocessing**
• **tuning parameters:** n_estimators is equal : 3 Accuracy: 0.76 (+/- 0.01) n_estimators is equal : 5 Accuracy: 0.76 (+/- 0.00) n_estimators is equal : 7 Accuracy: 0.76 (+/- 0.01) n_estimators is equal : 9 Accuracy: 0.76 (+/- 0.02) n_estimators is equal : 11 Accuracy: 0.76 (+/- 0.01) n_estimators is equal : 13 Accuracy: 0.76 (+/- 0.01) n_estimators is equal : 15 Accuracy: 0.76 (+/- 0.01) n_estimators is equal : 17 Accuracy: 0.77 (+/- 0.01) n_estimators is equal : 19 Accuracy: 0.77 (+/- 0.01) n_estimators is equal : 21 Accuracy: 0.77 (+/- 0.01) n_estimators is equal : 23 Accuracy: 0.76 (+/- 0.01) n_estimators is equal : 25 Accuracy: 0.76 (+/- 0.01) n_estimators is equal : 27 Accuracy: 0.76 (+/- 0.01) n_estimators is equal : 29 Accuracy: 0.76 (+/- 0.01) n_estimators is equal : 31 Accuracy: 0.76 (+/- 0.01) n_estimators is equal : 33 Accuracy: 0.76 (+/- 0.01) n_estimators is equal : 35 Accuracy: 0.76 (+/- 0.01) n_estimators is equal : 37 Accuracy: 0.76 (+/- 0.02) n_estimators is equal : 39	• **tuning parameters:** n_estimators is equal : 3 Accuracy: 0.76 (+/- 0.01) n_estimators is equal : 5 Accuracy: 0.77 (+/- 0.02) n_estimators is equal : 7 Accuracy: 0.78 (+/- 0.02) n_estimators is equal : 9 Accuracy: 0.78 (+/- 0.02) n_estimators is equal : 11 Accuracy: 0.78 (+/- 0.02) n_estimators is equal : 13 Accuracy: 0.78 (+/- 0.02) n_estimators is equal : 15 Accuracy: 0.78 (+/- 0.02) n_estimators is equal : 17 Accuracy: 0.78 (+/- 0.02) n_estimators is equal : 19 Accuracy: 0.78 (+/- 0.01) n_estimators is equal : 21 Accuracy: 0.78 (+/- 0.02) n_estimators is equal : 23 Accuracy: 0.78 (+/- 0.02) n_estimators is equal : 25 Accuracy: 0.78 (+/- 0.01) n_estimators is equal : 27 Accuracy: 0.78 (+/- 0.01) n_estimators is equal : 29 Accuracy: 0.78 (+/- 0.01) n_estimators is equal : 31 Accuracy: 0.78 (+/- 0.01) n_estimators is equal : 33 Accuracy: 0.78 (+/- 0.01) n_estimators is equal : 35 Accuracy: 0.78 (+/- 0.01) n_estimators is equal : 37 Accuracy: 0.78 (+/- 0.01) n_estimators is equal : 39

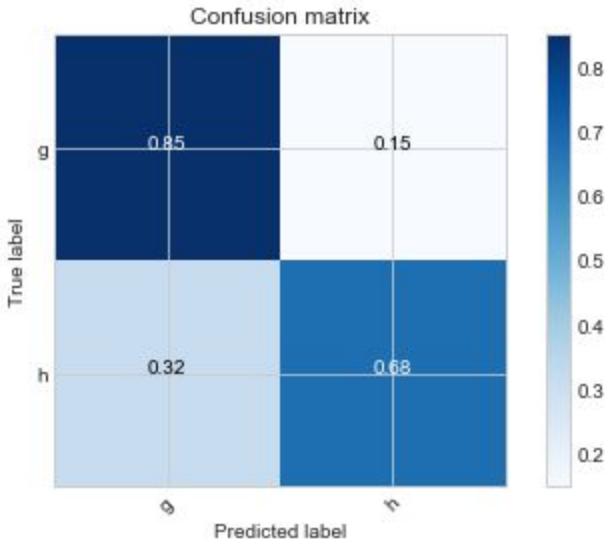
Accuracy: 0.76 (+/- 0.01)
n_estimators is equal : 41
Accuracy: 0.76 (+/- 0.01)
n_estimators is equal : 43
Accuracy: 0.76 (+/- 0.01)
n_estimators is equal : 45
Accuracy: 0.76 (+/- 0.01)
n_estimators is equal : 47
Accuracy: 0.76 (+/- 0.01)
n_estimators is equal : 49
Accuracy: 0.76 (+/- 0.01)

n estimators =17 with accuracy 77% on training set and 76.5% on test set

testing accuracy is : 0.7655120857214054
precision_score is : 0.7733610850110568
recall_score is : 0.7655120857214054
F Measure is : 0.7638519362147482

Normalized confusion matrix

[[0.8498004 0.1501996]
[0.31856645 0.68143355]]



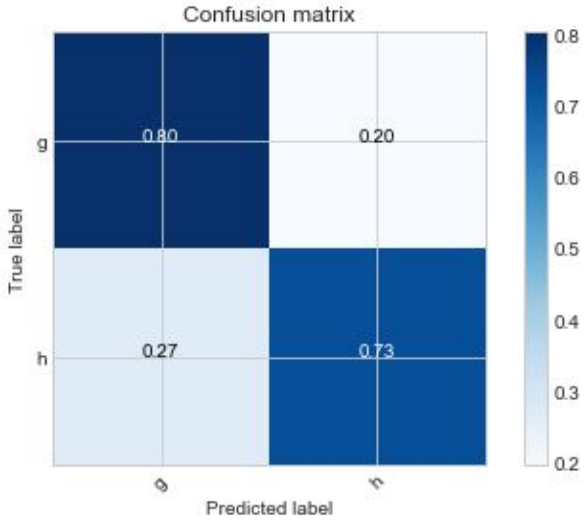
Accuracy: 0.78 (+/- 0.02)
n_estimators is equal : 41
Accuracy: 0.78 (+/- 0.02)
n_estimators is equal : 43
Accuracy: 0.78 (+/- 0.02)
n_estimators is equal : 45
Accuracy: 0.78 (+/- 0.01)
n_estimators is equal : 47
Accuracy: 0.78 (+/- 0.02)
n_estimators is equal : 49
Accuracy: 0.78 (+/- 0.02)

n estimators =9 with accuracy 78% on training set and 76.7% on test set

testing accuracy is : 0.7667580363817593
precision_score is : 0.7684935764644514
recall_score is : 0.7667580363817593
F Measure is : 0.7664923813102846

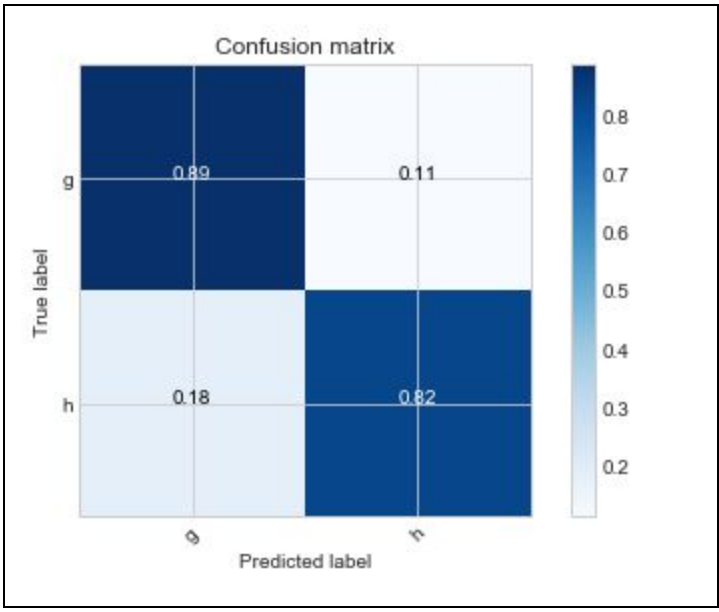
Normalized confusion matrix

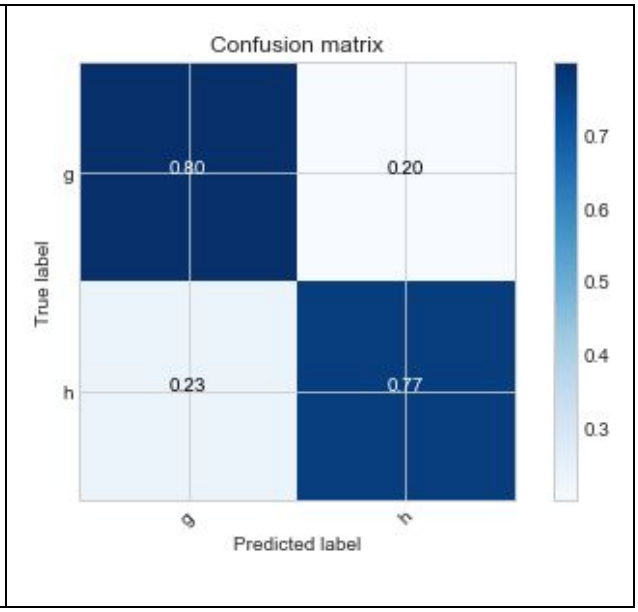
[[0.8034188 0.1965812]
[0.26926877 0.73073123]]



- **Random Forests**
  The random forests is trained the exact same way AdaBoost is trained earlier. It can be seen that random forests achieves maximum accuracy of 85 % at n_estimators = 120, and takes 18.8s to fit and predict data, which is better than AdaBoost in terms of both accuracy (82%) and time (21s).

before preprocessing	**after preprocessing**
● **tuning parameters** n_estimators is equal : 10 Accuracy: 0.84 (+/- 0.01) n_estimators is equal : 120 Accuracy: 0.85 (+/- 0.02) n_estimators is equal : 230 Accuracy: 0.85 (+/- 0.02) n_estimators is equal : 340 Accuracy: 0.85 (+/- 0.01) n_estimators is equal : 450 Accuracy: 0.85 (+/- 0.01) n_estimators is equal : 560 Accuracy: 0.85 (+/- 0.01) n_estimators is equal : 670 Accuracy: 0.85 (+/- 0.01) n_estimators is equal : 780 Accuracy: 0.85 (+/- 0.01) n_estimators is equal : 890 Accuracy: 0.85 (+/- 0.01) n_estimators is equal : 1000 Accuracy: 0.85 (+/- 0.01)	● **tuning parameters** n_estimators is equal : 10 Accuracy: 0.77 (+/- 0.01) n_estimators is equal : 120 Accuracy: 0.79 (+/- 0.02) n_estimators is equal : 230 Accuracy: 0.79 (+/- 0.01) n_estimators is equal : 340 Accuracy: 0.79 (+/- 0.01) n_estimators is equal : 450 Accuracy: 0.79 (+/- 0.02) n_estimators is equal : 560 Accuracy: 0.79 (+/- 0.02) n_estimators is equal : 670 Accuracy: 0.79 (+/- 0.02) n_estimators is equal : 780 Accuracy: 0.79 (+/- 0.01) n_estimators is equal : 890 Accuracy: 0.79 (+/- 0.01) n_estimators is equal : 1000 Accuracy: 0.79 (+/- 0.01)
● **n estimators =50 with accuracy 85% on training set and 85% on test set** testing accuracy is : 0.8517318714178919 precision_score is : 0.8533435407239559 recall_score is : 0.8517318714178919 F Measure is : 0.8515701220459745	● **n estimators =50 with accuracy 85% on training set and 85% on test set** testing accuracy is : 0.7817094443060055 precision_score is : 0.7821408131354614 recall_score is : 0.7817094443060055 F Measure is : 0.7816733309567513
● **Normalized confusion matrix** [[0.88522954 0.11477046] [0.18168243 0.81831757]]	● **Normalized confusion matrix** [[0.79788839 0.20211161] [0.23418972 0.76581028]]





- **SVM**
  - The Linear SVM classifier performed badly with accuracy less than 50% which means that the data was not linearly separable. The wide range of C regularization values did not introduce big contribution to the accuracy, which endorses the previous claim.
  - Also the initial graph of the unprocessed data shows from the beginning that SVM has the worst behavior
  - 0.5083478694243708 at C = 1 time = 17.718008041381836
  - 0.5058559681036631 at C = 5 time = 22.195687294006348
  - 0.5058559681036631 at C = 10 time = 21.83406376838684
  - 0.5058559681036631 at C = 100 time = 20.759934425354004
  - 0.5058559681036631 at C = 1000 time = 20.93567705154419

Conclusion:

Random Forests and AdaBoost have best performances

- **for preprocessed data**

Random Forests: 85% on training set and 85% on test set , F =78.2
KNN : accuracy 78% on training set and 76.7% on test set,F= 76.7
AdaBoost: accuracy 82% on training set and 76% on test set, F= 76

- **for not processed data:**

Random Forests: 85% on training set and 85% on test set , F =85.15
AdaBoost: accuracy 86.2% on training set and 82 % on test set, F =81.9
Decision Tree: accuracy 79.5 % on test set, F= 79.5%

After preprocessing comparison on test set

Before preprocessing comparison on test set