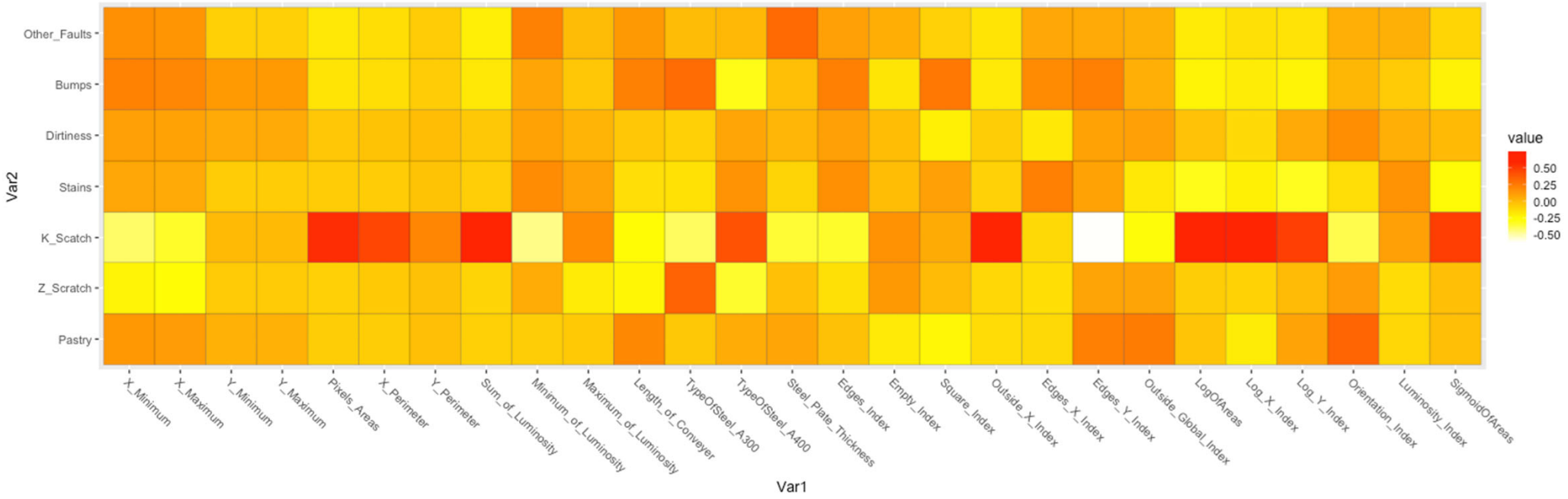
SDSC3006 Project
Classification of Steel Plate Faults

Data Exploration

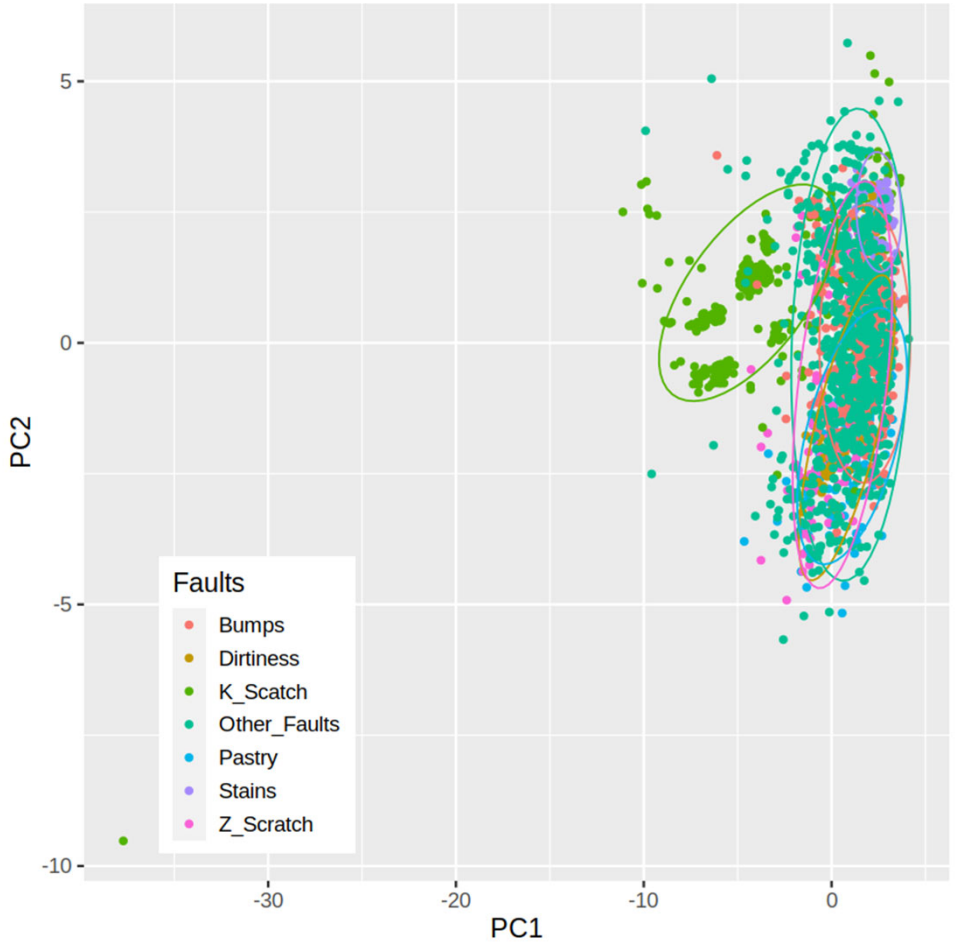
- 27 features
- 7 types of steel plate defects (present with One-Hot Encoding)
- The steel plate dataset can be download from the "UCI Machine Learning Repository".



Data Processing

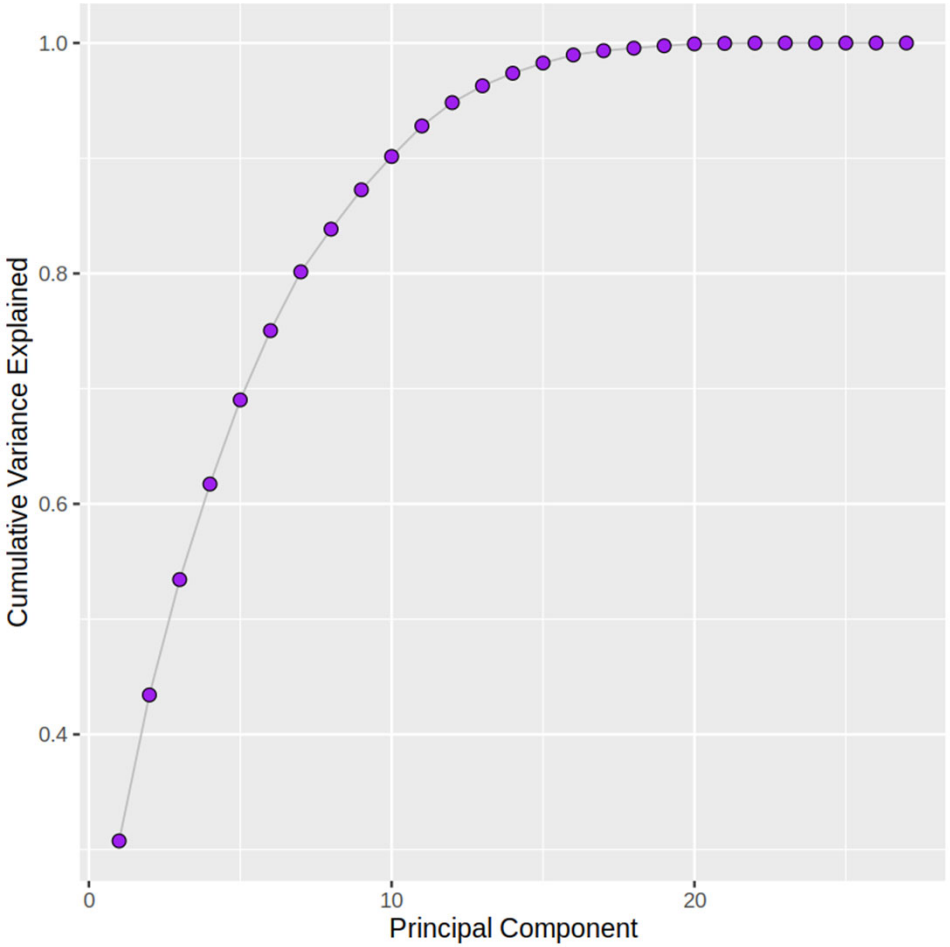
- Too many dimensions for the data set
- Distribution of the data are unknown

- Reduce 27 dimensions to 2 dimensions for visualization
- K_Scratch is more distinguished from other faults



Data Pre-processing (PCA)

- Reduce 27 dimensions to 12 dimensions
- N = 12 accounts for 95% variance of the data



Goal

- Obser which method is better for this data
- Obser if PCA helps with increase the classification accuracy

Data Pre-processing (Train-Test Split)

- Randomly sample 80% of the data as training set, and the remaining as the test set

```
set.seed(seed)
rand <- sample(nrow(df), nrow(df) * 0.8)

df.train.X <- df.X[rand, ]
df.train.Y <- df.Y[rand, ]
df.train.pcaX <- df.pcaX[rand, ]

df.test.X <- df.X[-rand, ]
df.test.Y <- df.Y[-rand, ]
df.test.pcaX <- df.pcaX[-rand, ]
```

Methods

All methods were implemented with and without PCA

1)Naive Bayes Classifier

2)Multinomial Logistic Regression

3)Random Forest

4)K-Nearest Neighbour

5)Boosting

6)Learning Vector Quantization

7)Support Vector Machine

    1)Linear

    2)Polynomial

    3)Radial

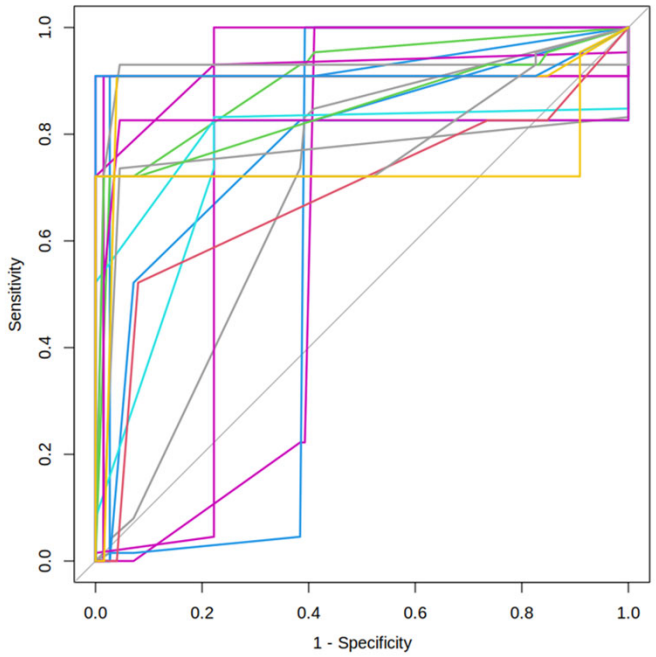
Justification

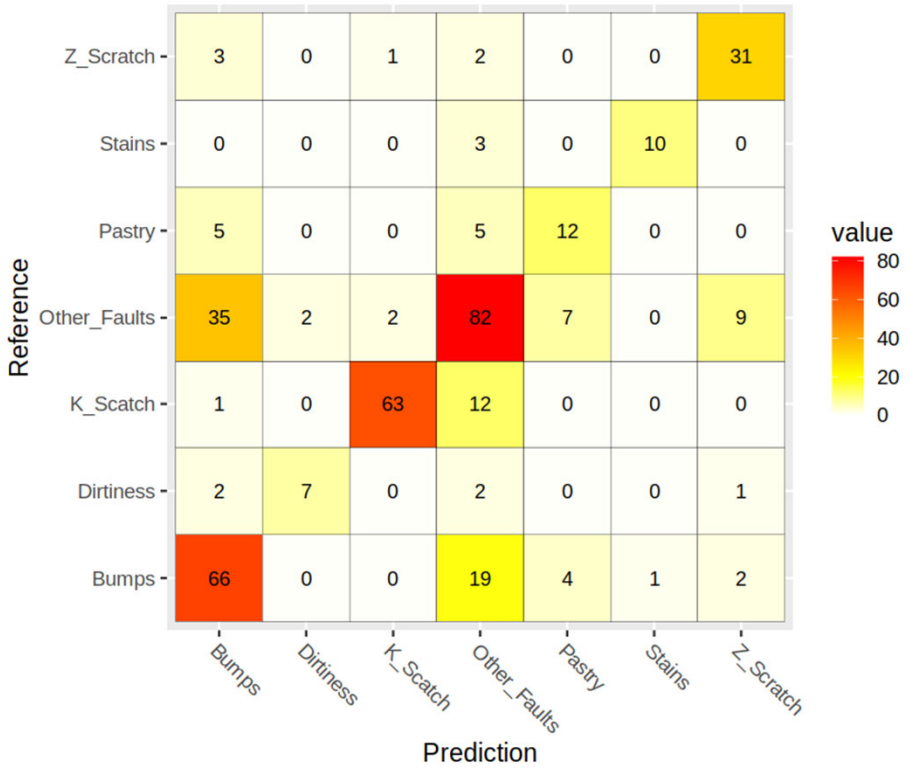
Effect of the method of PCA was unknown.

Test if PCA helps.
Observe which method is the best, with or without PCA

Naive Bayes Classifier (with PCA)

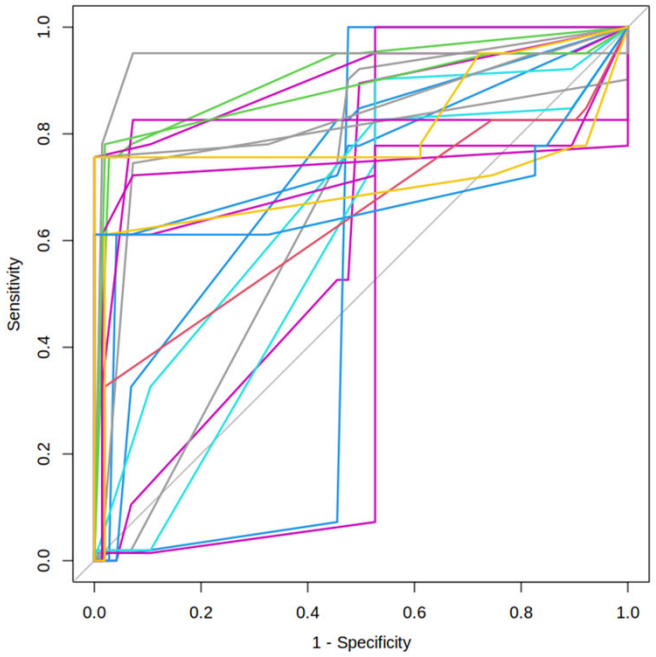
- Accuracy on training set = 0.700
- Accuracy on test test = 0.697

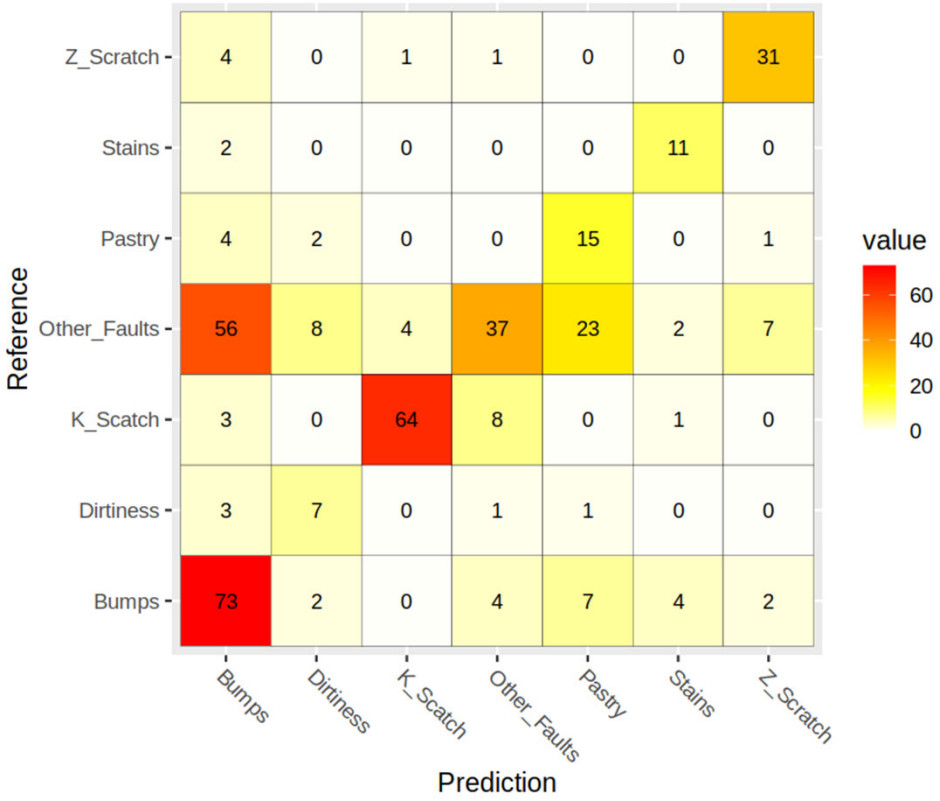




Naive Bayes Classifier (without PCA)

- Accuracy on training set = 0.610
- Accuracy on test test = 0.612

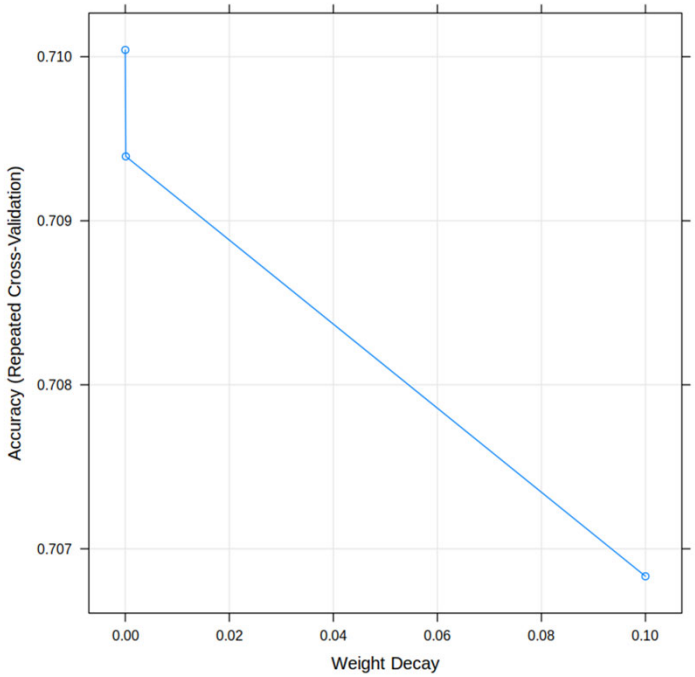




Multinomial Logistic Regression (with PCA)

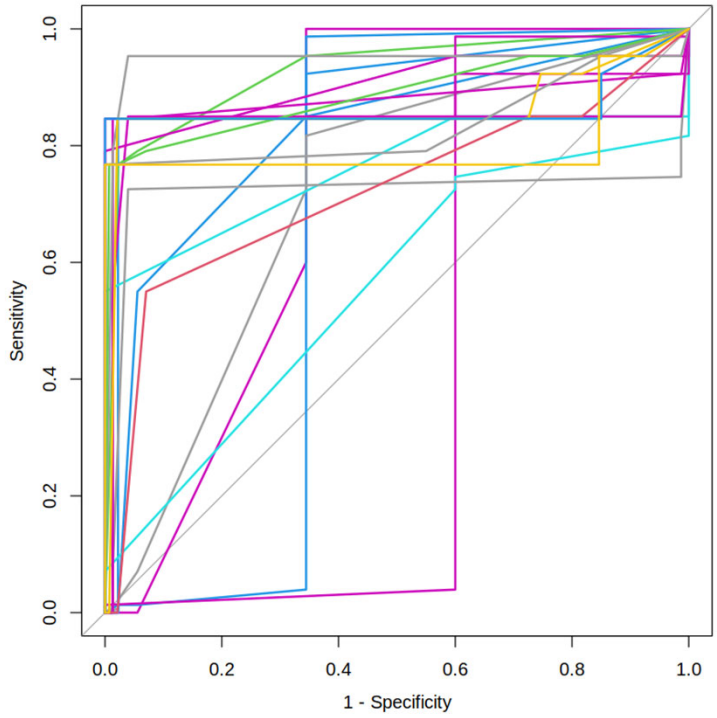
- Perform 10-fold Cross-Validation to find out the optimal decay value

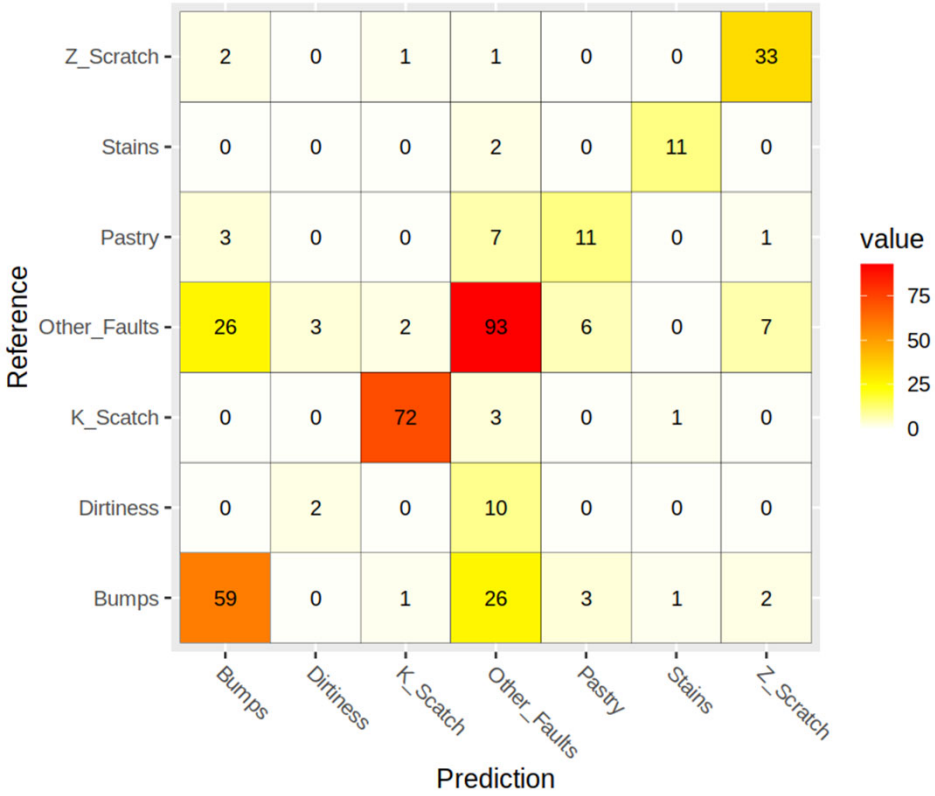
```
No pre-processing
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 1397, 1395, 1395, 1397, 1398, 1398, ...
Resampling results across tuning parameters:

  decay  Accuracy   Kappa
  0e+00  0.7100417  0.6258953
  1e-04  0.7093923  0.6250779
  1e-01  0.7068323  0.6217833

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was decay = 0.
```



Multinomial Logistic Regression (with PCA)

- Accuracy on training set = 0.719
- Accuracy on test test = 0.722

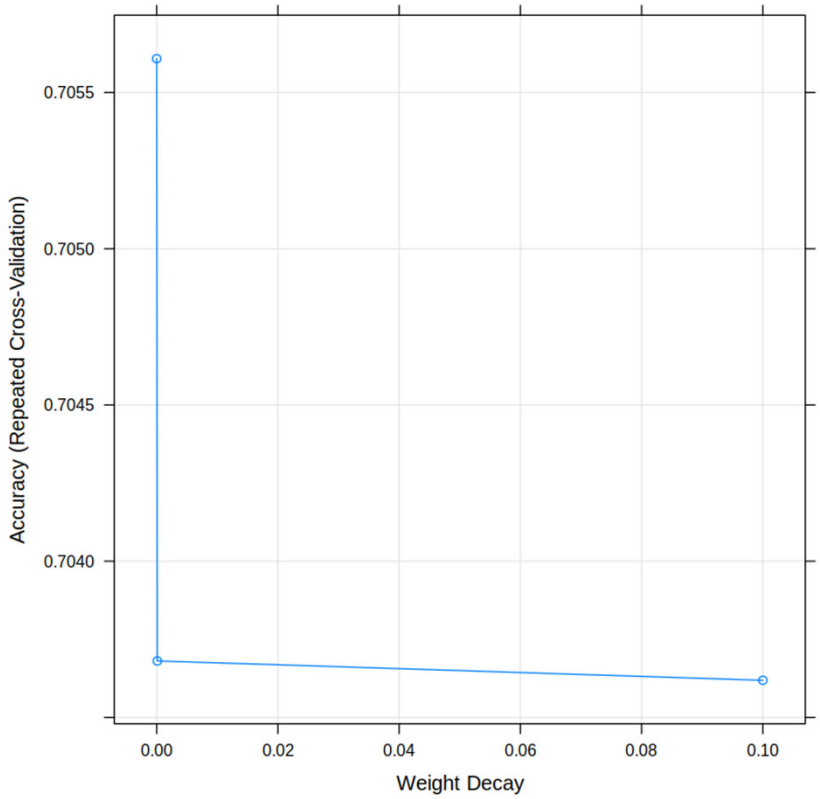




Multinomial Logistic Regression (without PCA)

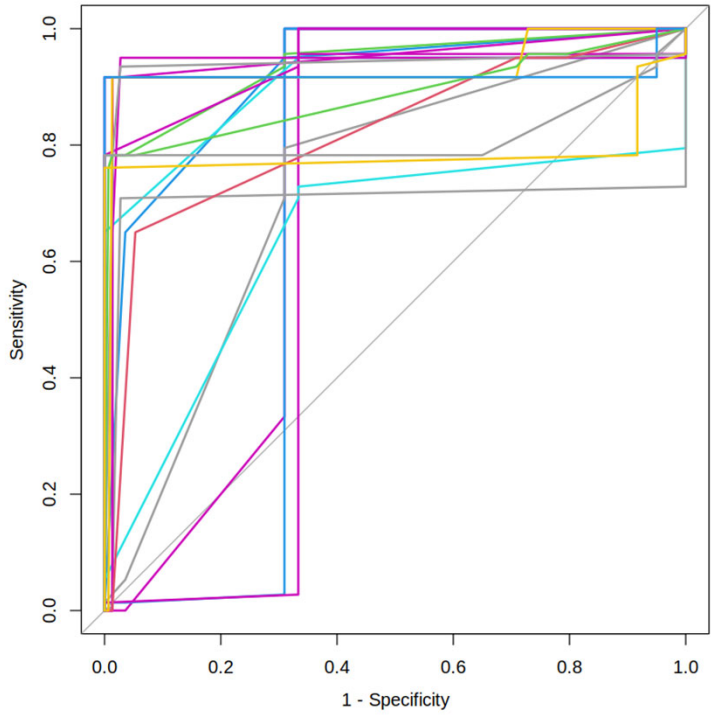
- Perform 10-fold Cross-Validation to find out the optimal decay value (decay = 0)

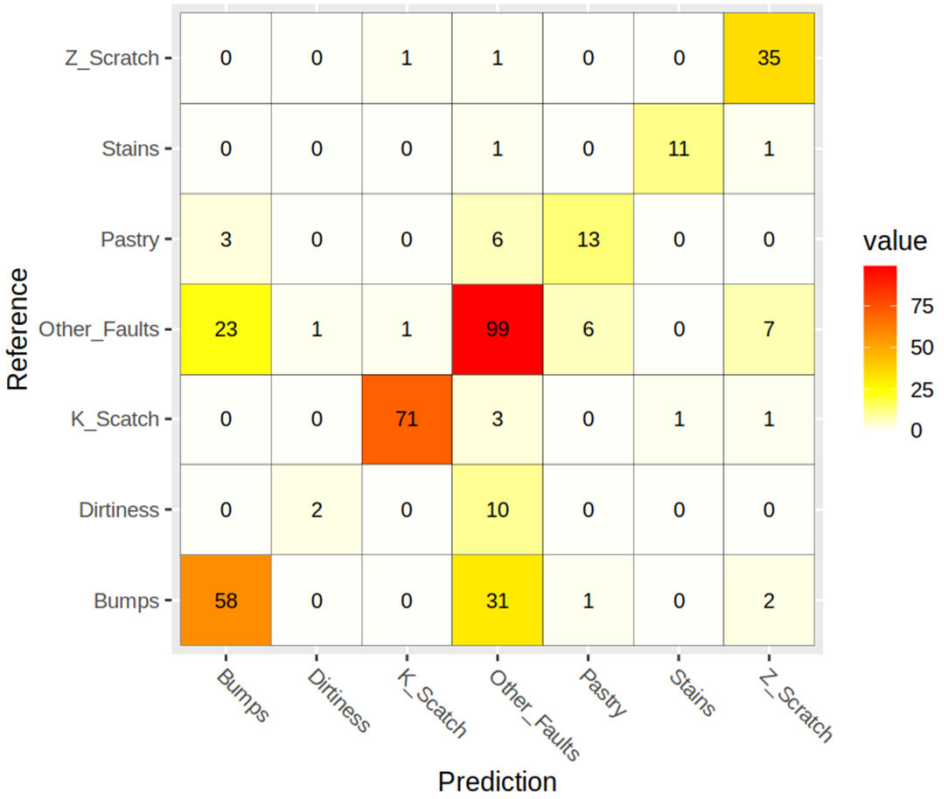
```
No pre-processing
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 1397, 1395, 1395, 1397, 1398, 1398, ...
Resampling results across tuning parameters:

  decay  Accuracy   Kappa
  0e+00  0.7056081  0.6215307
  1e-04  0.7036808  0.6193123
  1e-01  0.7036189  0.6186091

Accuracy was used to select the optimal model using the largest value
The final value used for the model was decay = 0.
```



Multinomial Logistic Regression (without PCA)

- Accuracy on training set = 0.740
- Accuracy on test test = 0.743

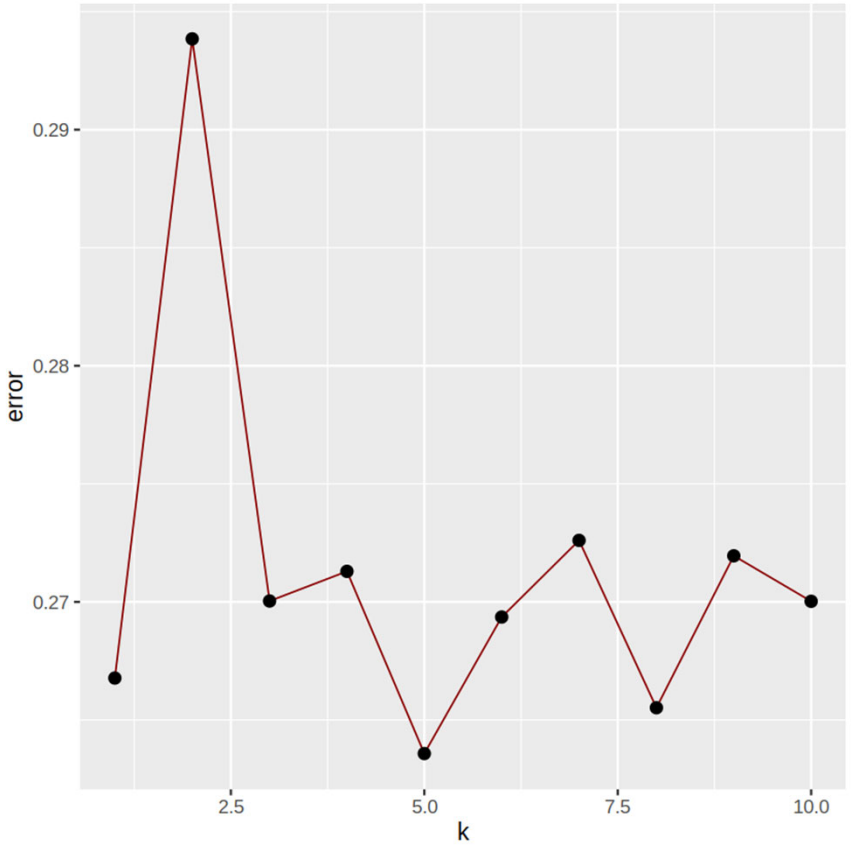




K-Nearest Neighbour (with PCA)

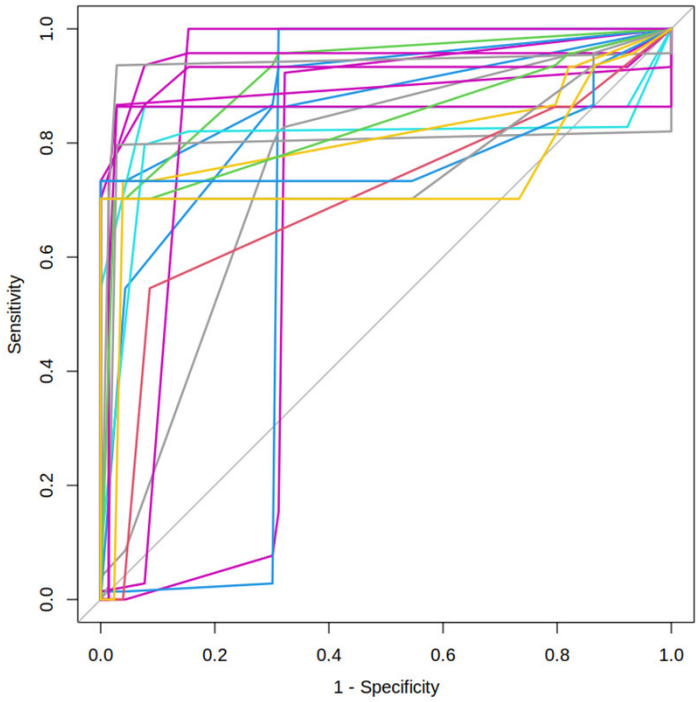
- Perform 10-fold Cross-Validation to find out the optimal k value (k = 5)

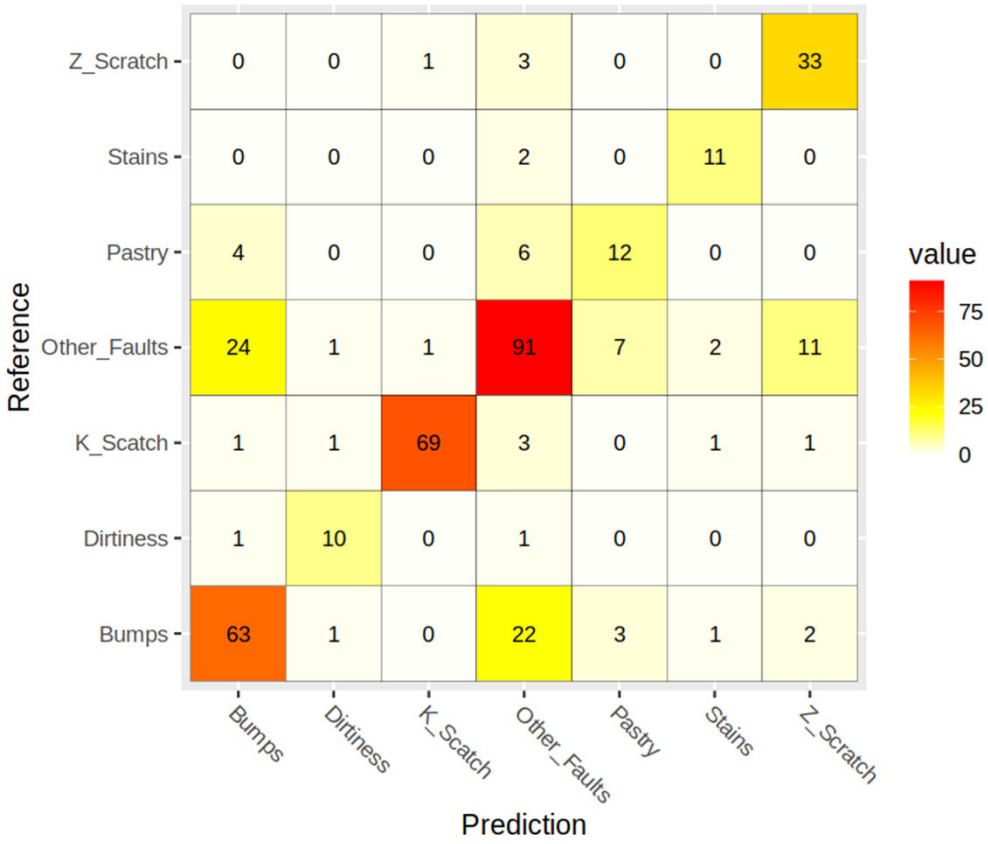
```
- best performance: 0.2635773

- Detailed performance results:
    k     error dispersion
1   1 0.2667742 0.03247097
2   2 0.2938462 0.03389177
3   3 0.2700372 0.04464668
4   4 0.2712945 0.02889204
5   5 0.2635773 0.02786094
6   6 0.2693590 0.02516199
7   7 0.2726055 0.02820199
8   8 0.2655170 0.02713810
9   9 0.2719520 0.02789193
10 10 0.2700248 0.02672261
```



K-Nearest Neighbour (with PCA)

- Accuracy on training set = 0.823
- Accuracy on test test = 0.743

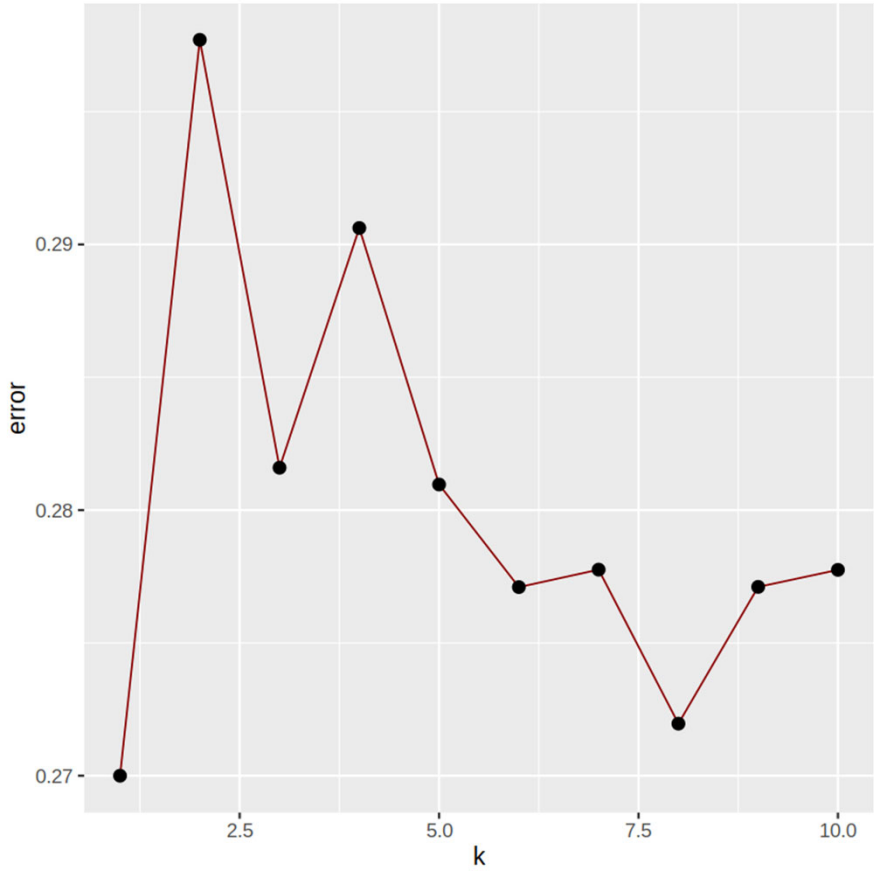




K-Nearest Neighbour (without PCA)

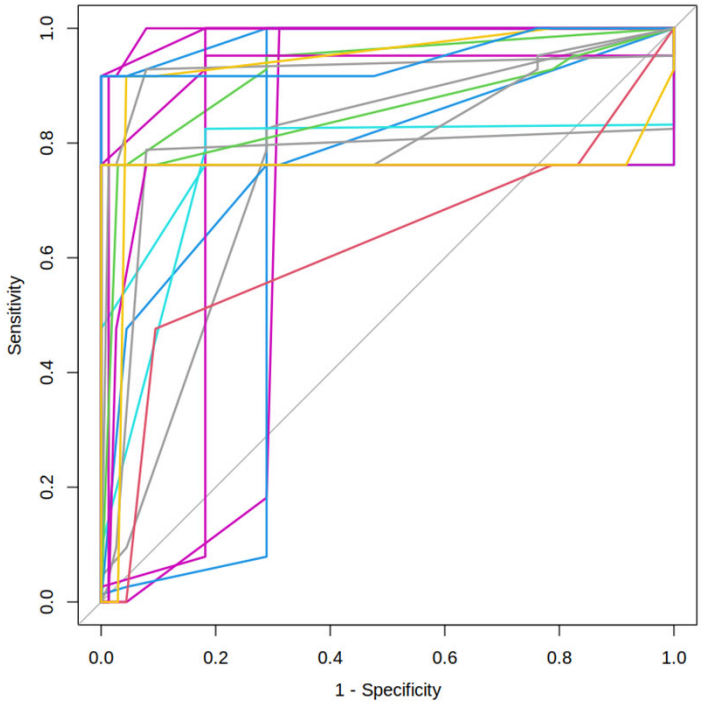
- Perform 10-fold Cross-Validation to find out the optimal k value (k = 1)

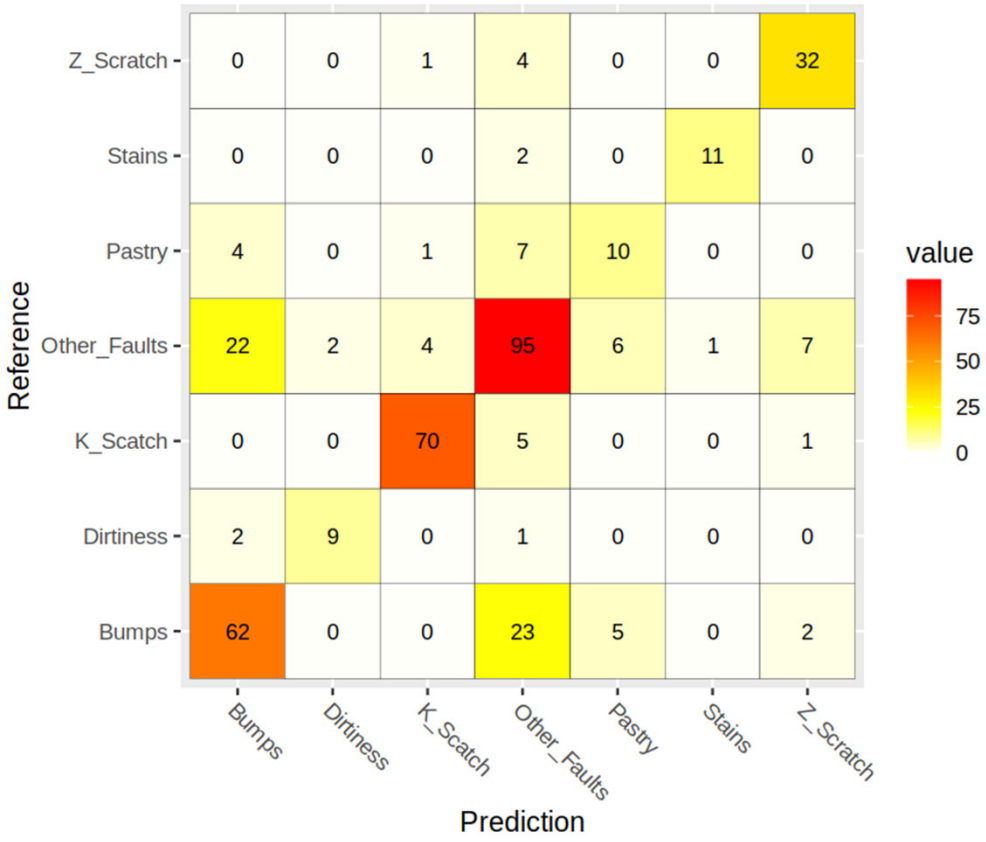
```
- best performance: 0.27

- Detailed performance results:
    k     error dispersion
1   1 0.2700000 0.03138458
2   2 0.2977047 0.02692294
3   3 0.2815964 0.02074165
4   4 0.2906203 0.02397171
5   5 0.2809636 0.02673920
6   6 0.2771009 0.02602793
7   7 0.2777585 0.03486108
8   8 0.2719603 0.03036039
9   9 0.2771092 0.03256651
10 10 0.2777502 0.03025606
```



K-Nearest Neighbour (without PCA)

- Accuracy on training set = 1
- Accuracy on test test = 0.743

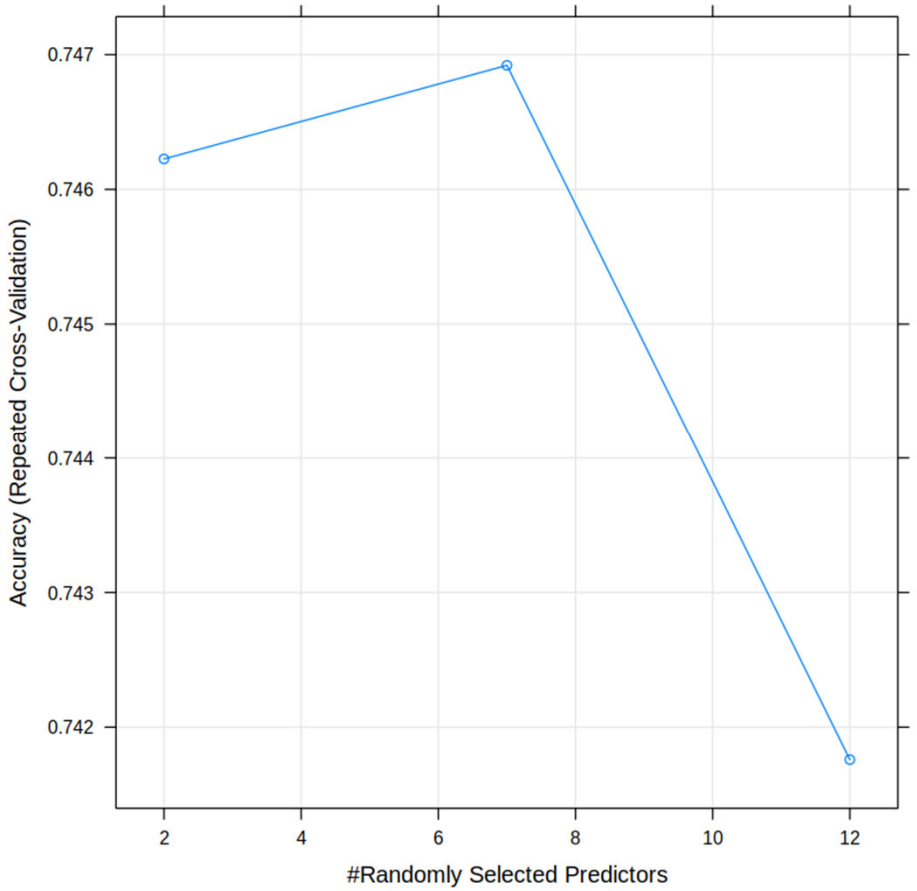




Random Forest (with PCA)

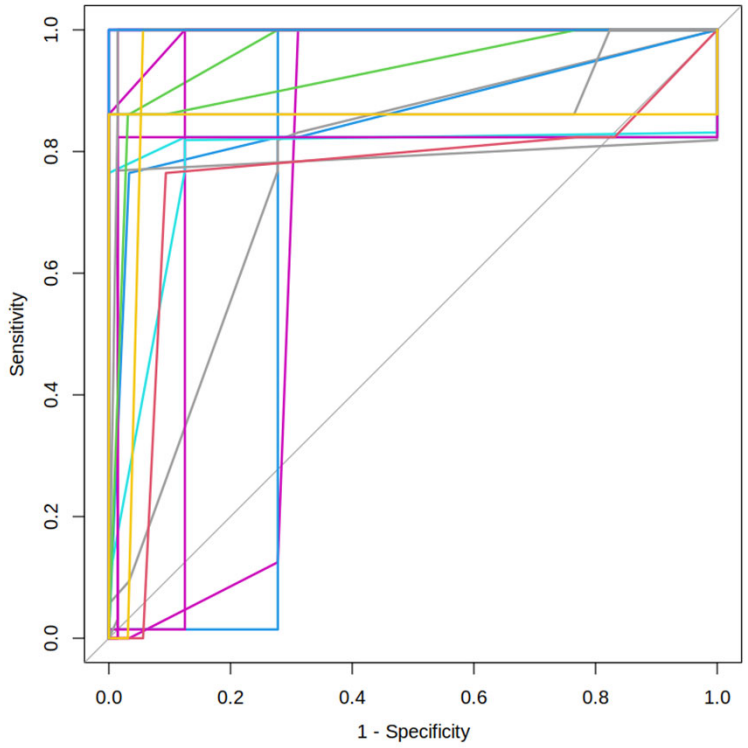
- Perform 10-fold Cross-Validation to find out the optimal mtry value
- (mtry = 7)

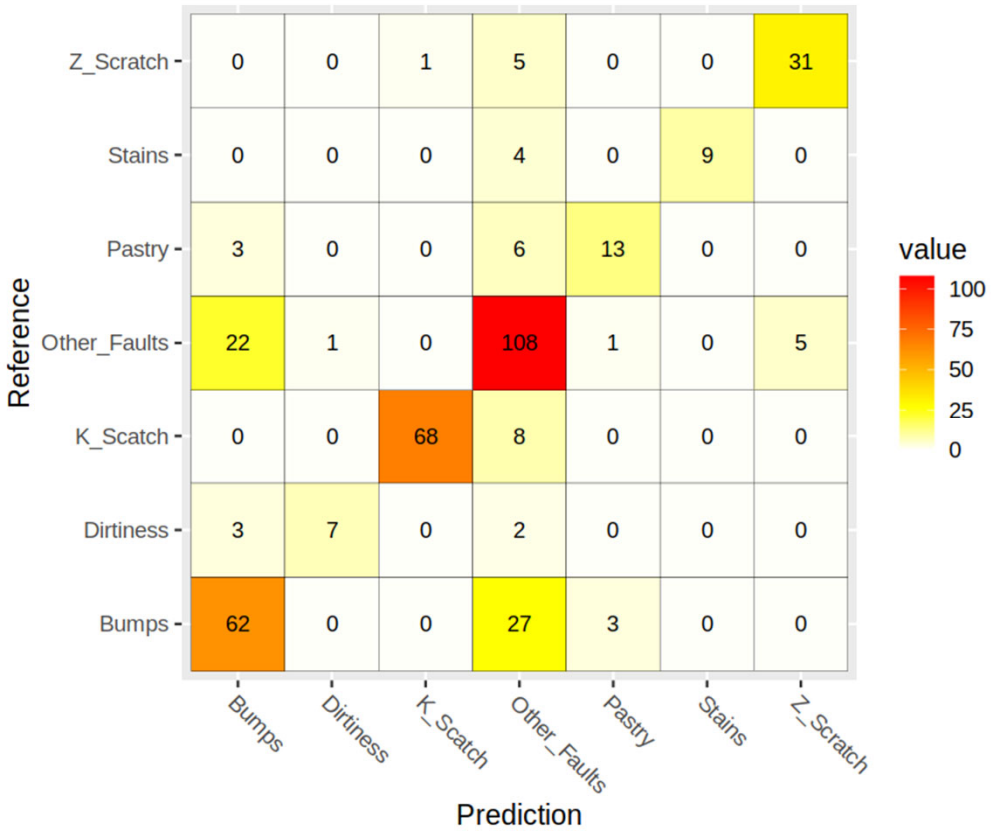
```
No pre-processing
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 1397, 1395, 1395, 1397, 1398, 1398, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.7462257  0.6686634
   7    0.7469205  0.6710535
  12    0.7417591  0.6643989

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 7.
```



Random Forest (with PCA)

- Accuracy on training set = 1
- Accuracy on test test = 0.766

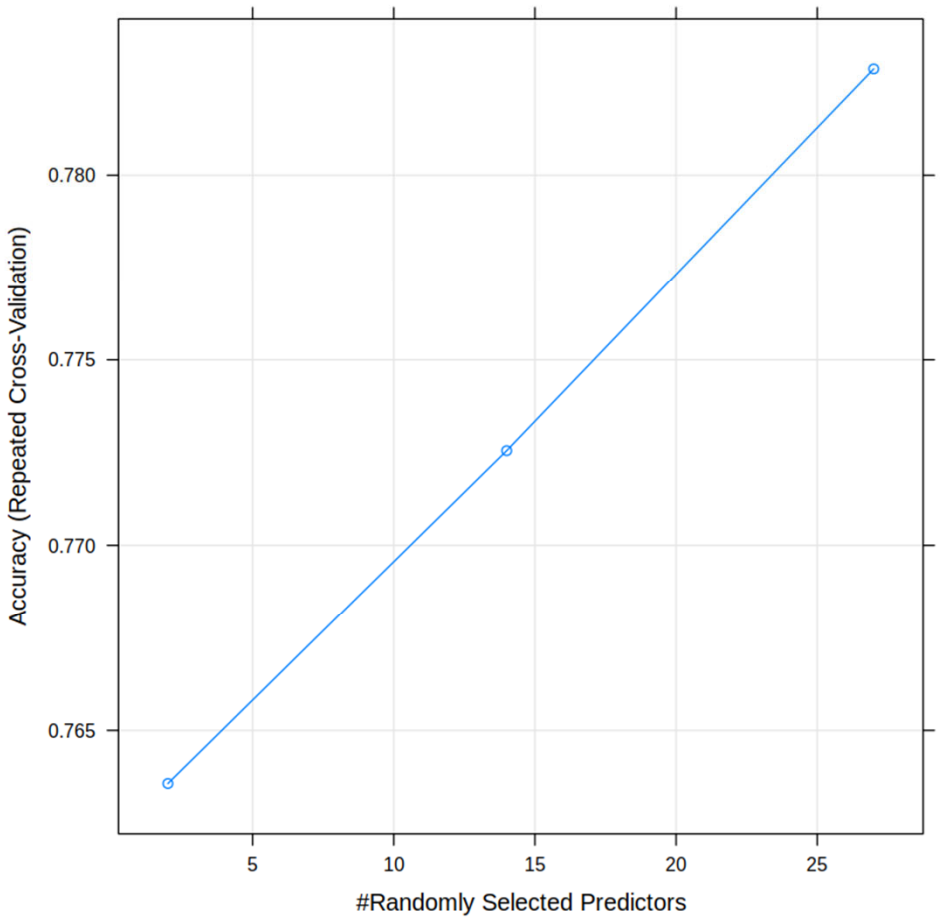




Random Forest (without PCA)

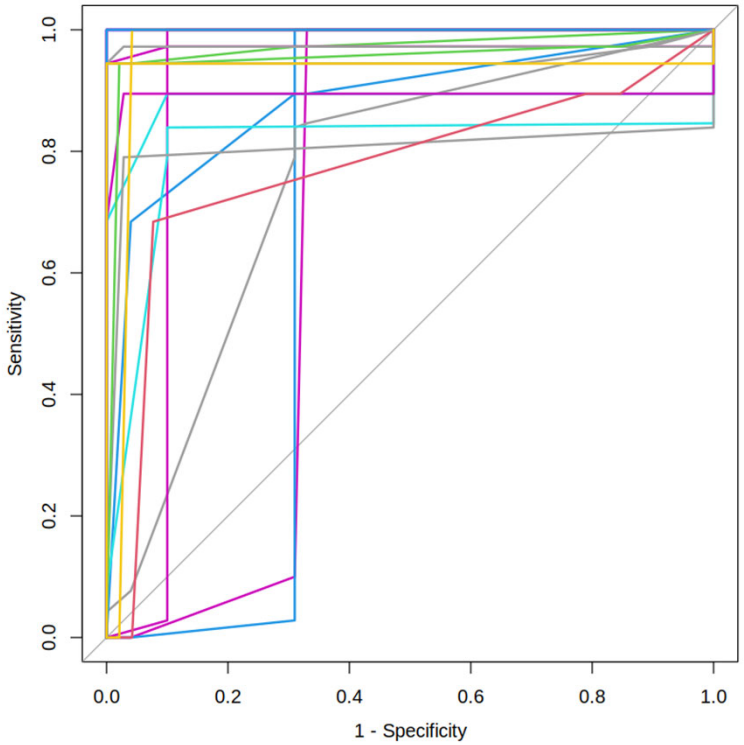
- Perform 10-fold Cross-Validation to find out the optimal mtry value
- (mtry = 27)

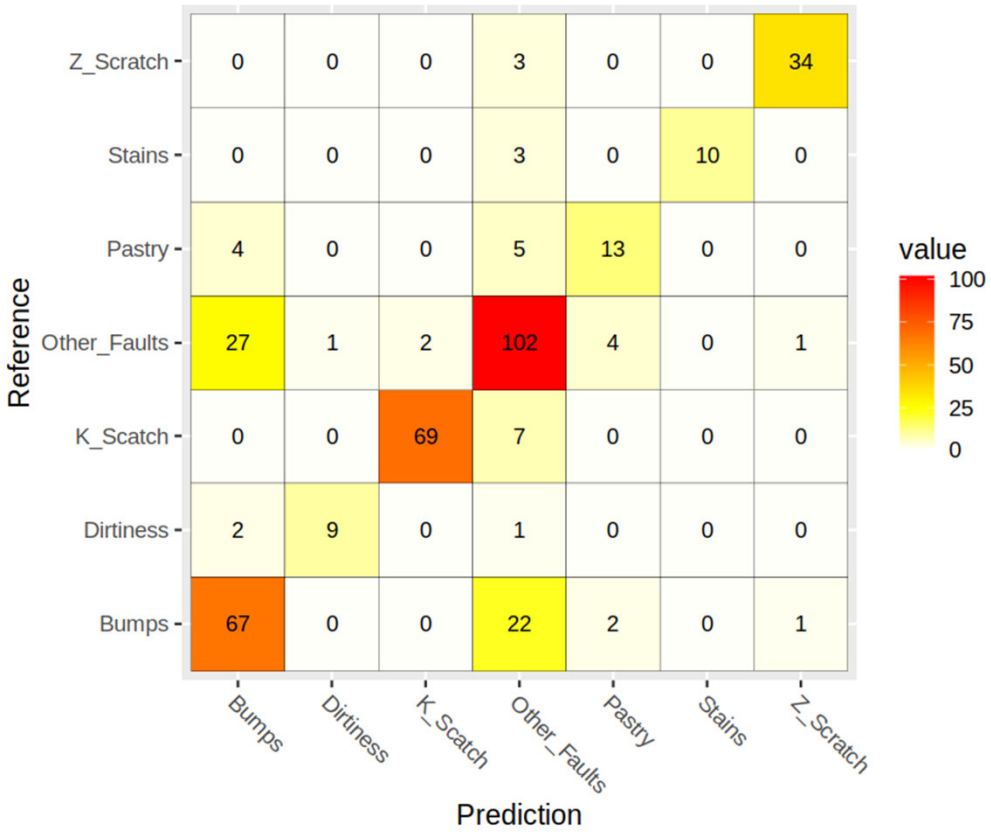
```
No pre-processing
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 1397, 1395, 1395, 1397, 1398, 1398, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.7635715  0.6920532
  14    0.7725504  0.7049538
  27    0.7828734  0.7185713

Accuracy was used to select the optimal model using the largest value
The final value used for the model was mtry = 27.
```



Random Forest (without PCA)

- Accuracy on training set = 1
- Accuracy on test test = 0.781



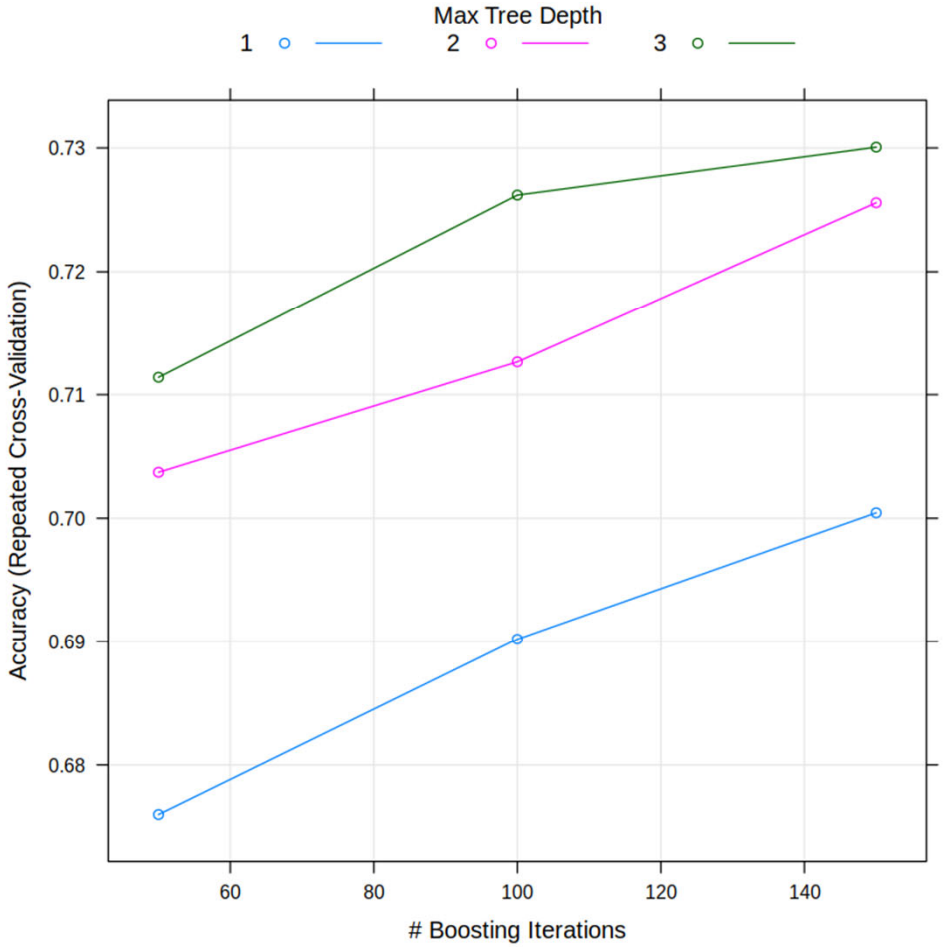


Boosting (with PCA)

- Perform 10-fold Cross-Validation to find out the optimal shrinkage value, number of trees, interaction depth and n.minobsinnode

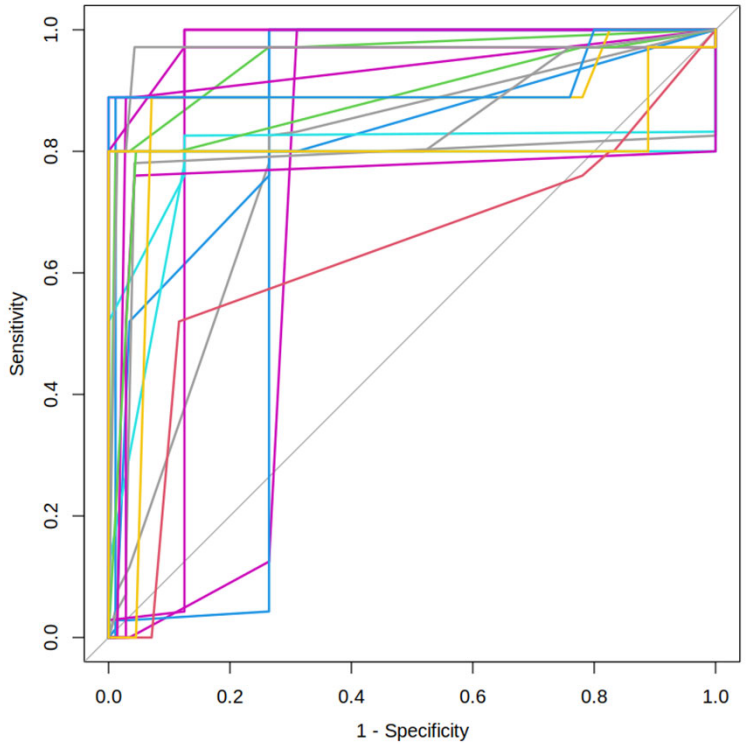
interaction.depth	n.trees	Accuracy	Kappa
1	50	0.6759875	0.5691389
1	100	0.6901857	0.5938445
1	150	0.7004422	0.6090604
2	50	0.7037221	0.6110292
2	100	0.7126595	0.6260801
2	150	0.7255799	0.6432262
3	50	0.7114105	0.6218916
3	100	0.7261960	0.6437407
3	150	0.7300630	0.6496915

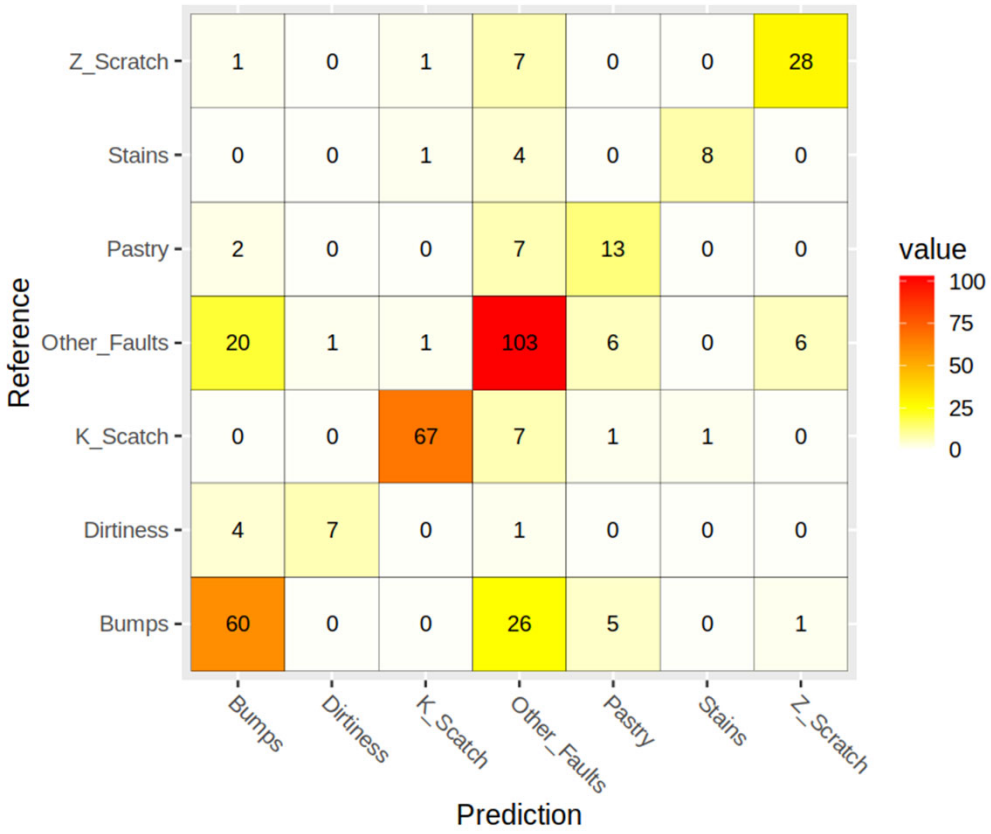
```
Tuning parameter 'shrinkage' was held constant at a value of 0.1

Tuning parameter 'n.minobsinnode' was held constant at a value of 10
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were n.trees = 150, interaction.depth =
 3, shrinkage = 0.1 and n.minobsinnode = 10.
```



Boosting (with PCA)

- Accuracy on training set = 0.933
- Accuracy on test test = 0.735



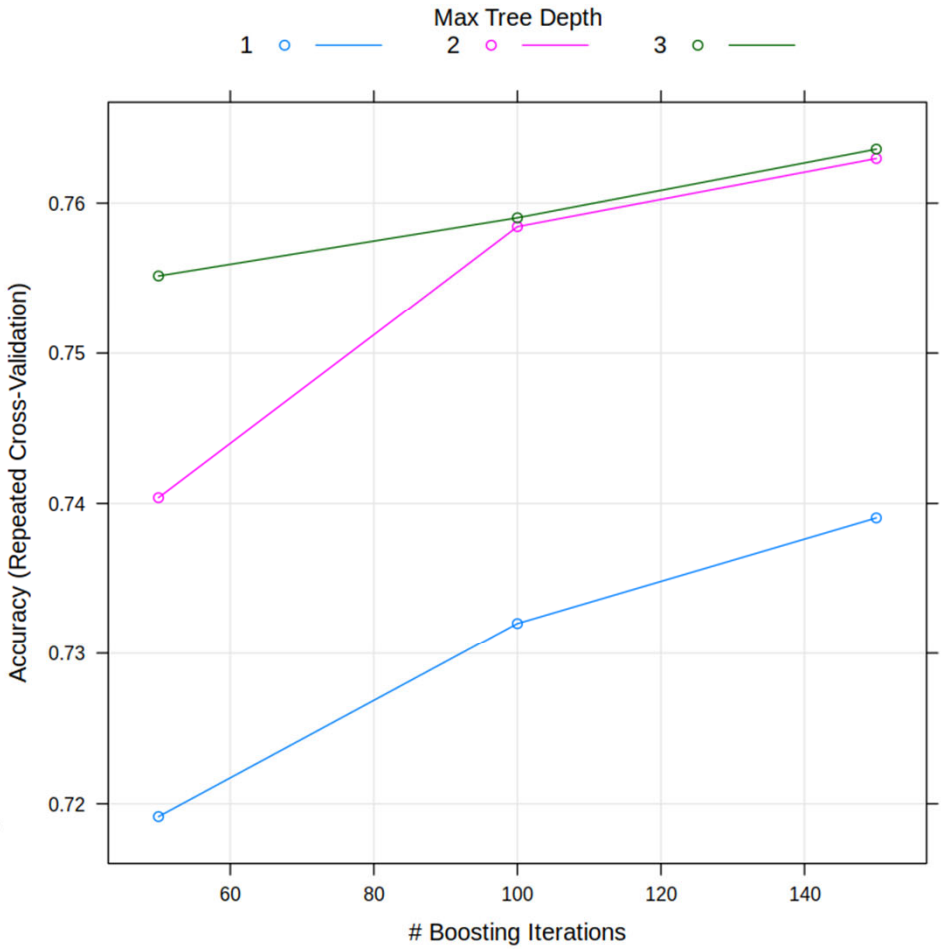


Boosting (without PCA)

- Perform 10-fold Cross-Validation to find out the optimal shrinkage value, number of trees, interaction depth and n.minobsinnode

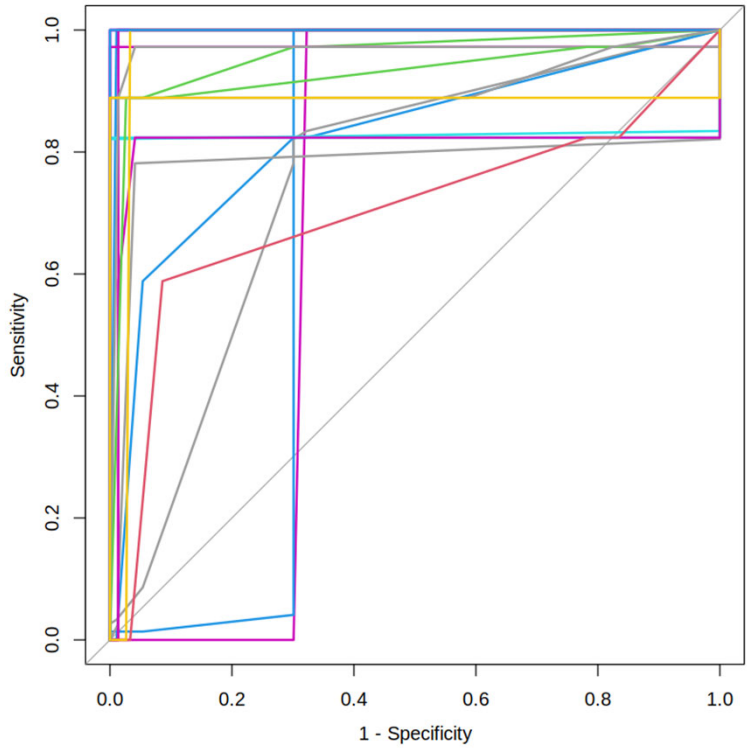
interaction.depth	n.trees	Accuracy	Kappa
1	50	0.7191488	0.6322553
1	100	0.7320031	0.6520644
1	150	0.7390380	0.6625470
2	50	0.7403824	0.6632255
2	100	0.7584475	0.6875999
2	150	0.7629639	0.6934045
3	50	0.7551756	0.6822838
3	100	0.7590344	0.6888130
3	150	0.7635839	0.6946416

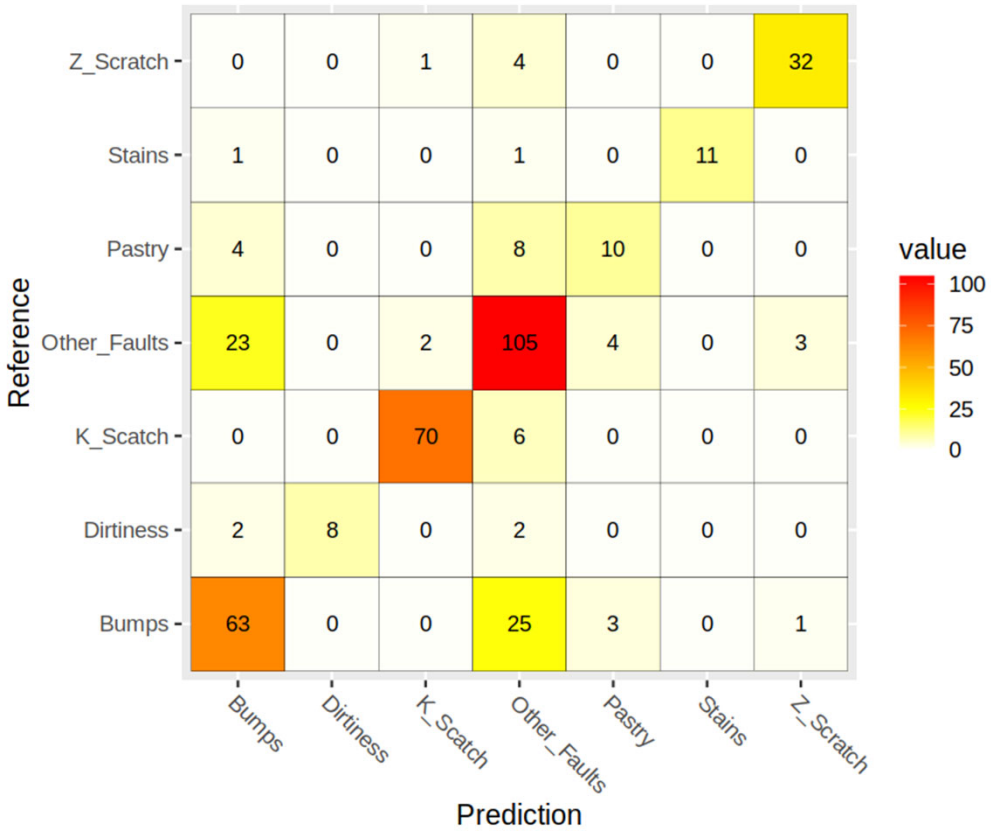
```
Tuning parameter 'shrinkage' was held constant at a value of 0.1

Tuning parameter 'n.minobsinnode' was held constant at a value of 10
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were n.trees = 150, interaction.depth =
 3, shrinkage = 0.1 and n.minobsinnode = 10.
```



Boosting (without PCA)

- Accuracy on training set = 0.957
- Accuracy on test test = 0.769

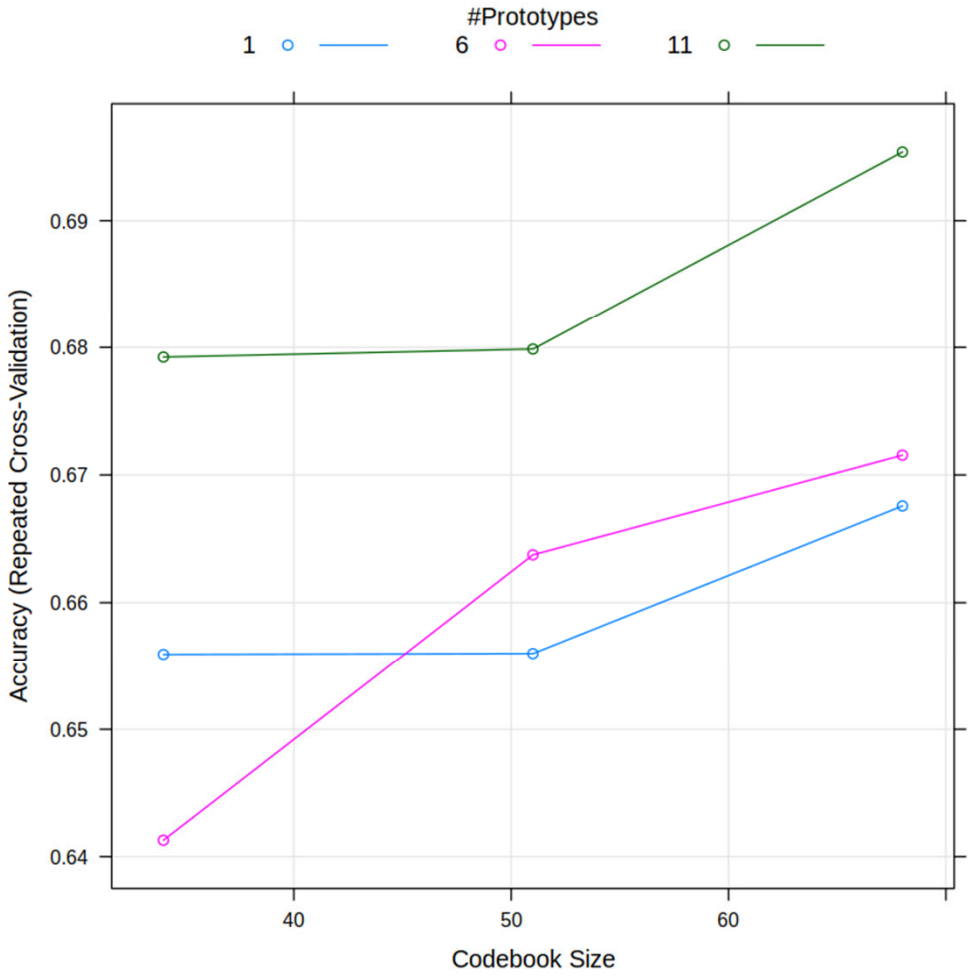




Learning Vector Quantization (with PCA)

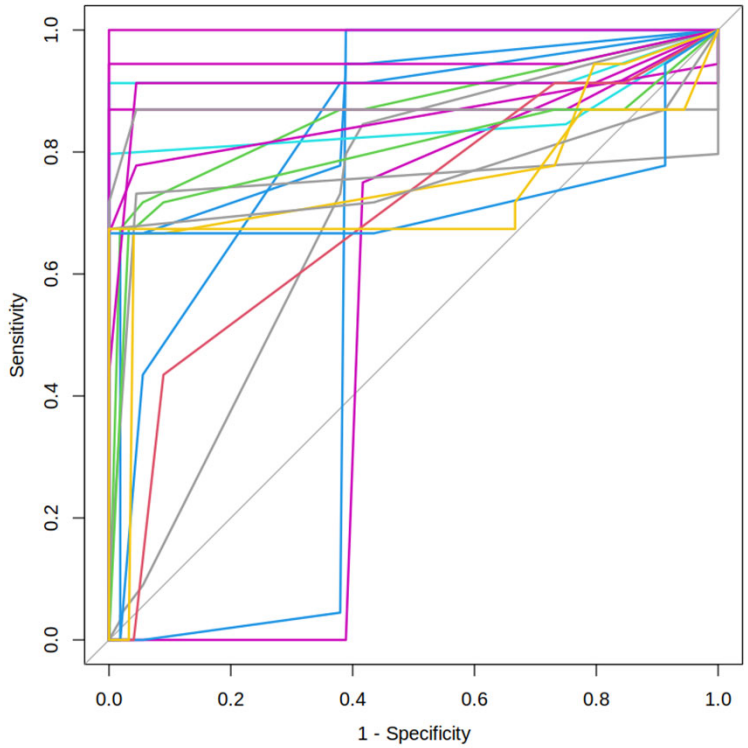
- Perform 10-fold Cross-Validation to find out the best size and k

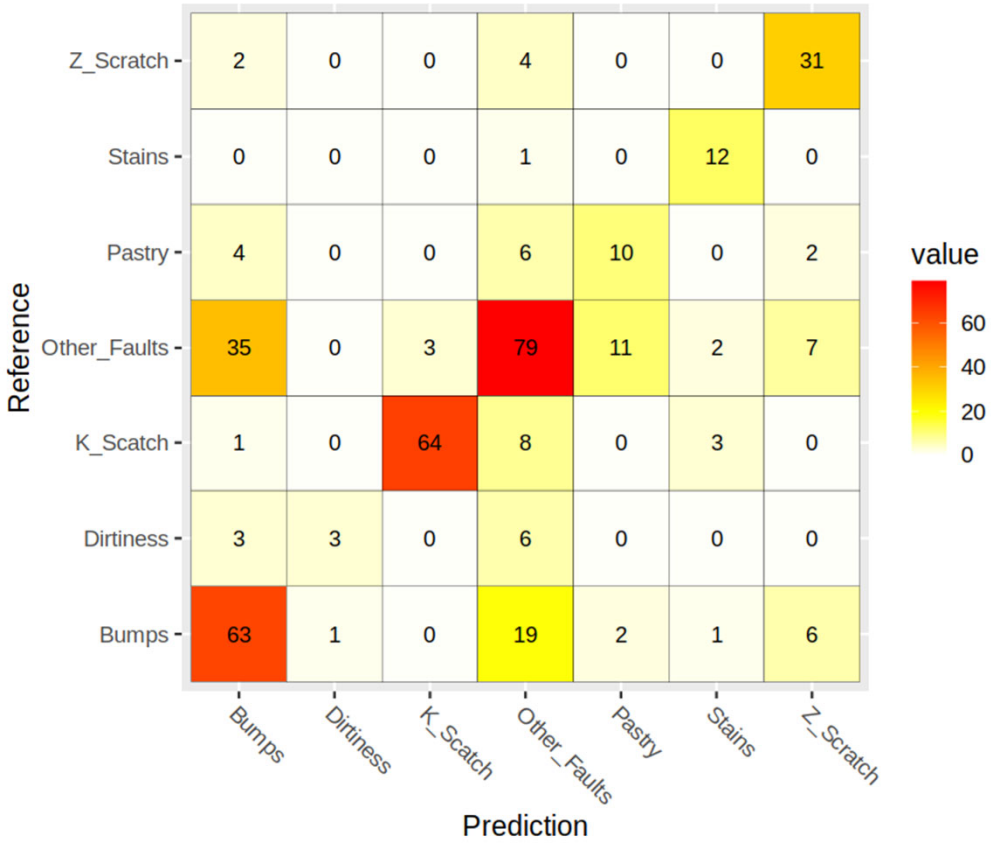
```
No pre-processing
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 1397, 1395, 1395, 1397, 1398, 1398, ...
Resampling results across tuning parameters:

  size  k   Accuracy   Kappa
  34     1  0.6559360  0.5618174
  34     6  0.6412872  0.5408845
  34    11  0.6792345  0.5901454
  51     1  0.6560027  0.5587813
  51     6  0.6637455  0.5698542
  51    11  0.6798627  0.5898471
  68     1  0.6675752  0.5748721
  68     6  0.6715496  0.5804535
  68    11  0.6953964  0.6093592

Accuracy was used to select the optimal model using the largest value
The final values used for the model were size = 68 and k = 11.
```



Learning Vector Quantization (with PCA)

- Accuracy on training set = 0.696
- Accuracy on test test = 0.674

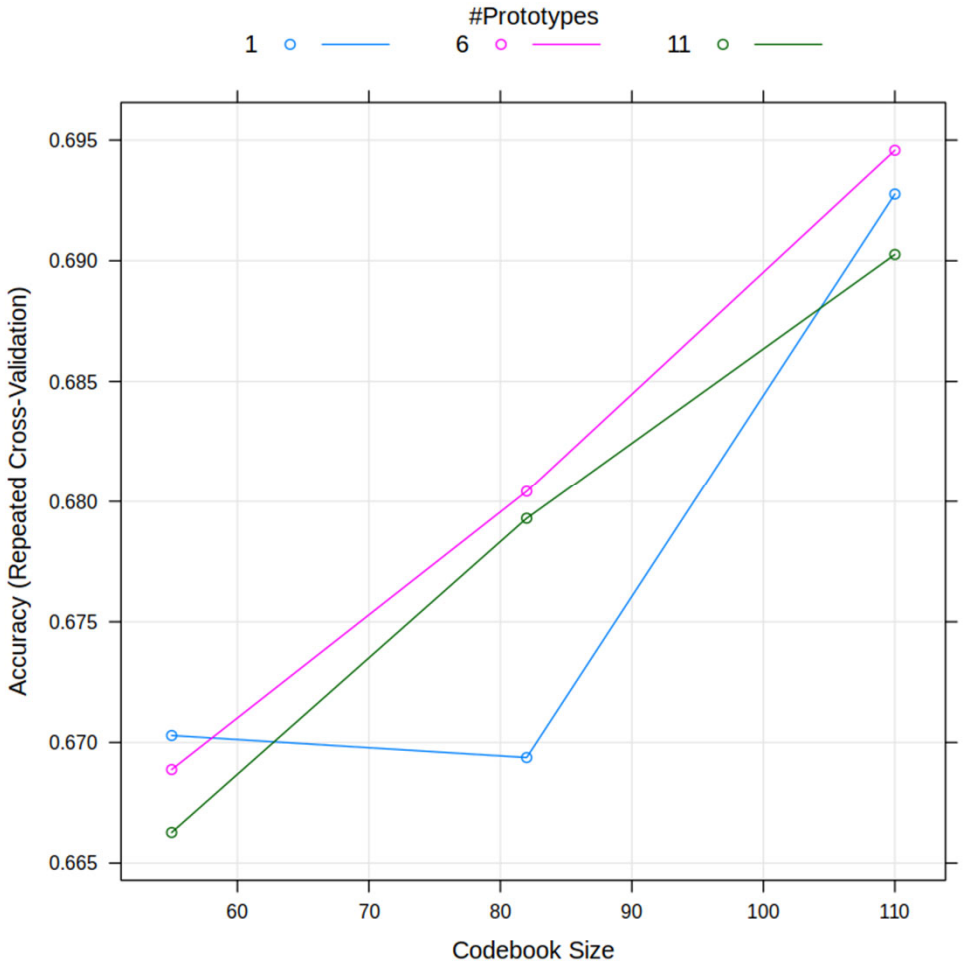




Learning Vector Quantization (without PCA)

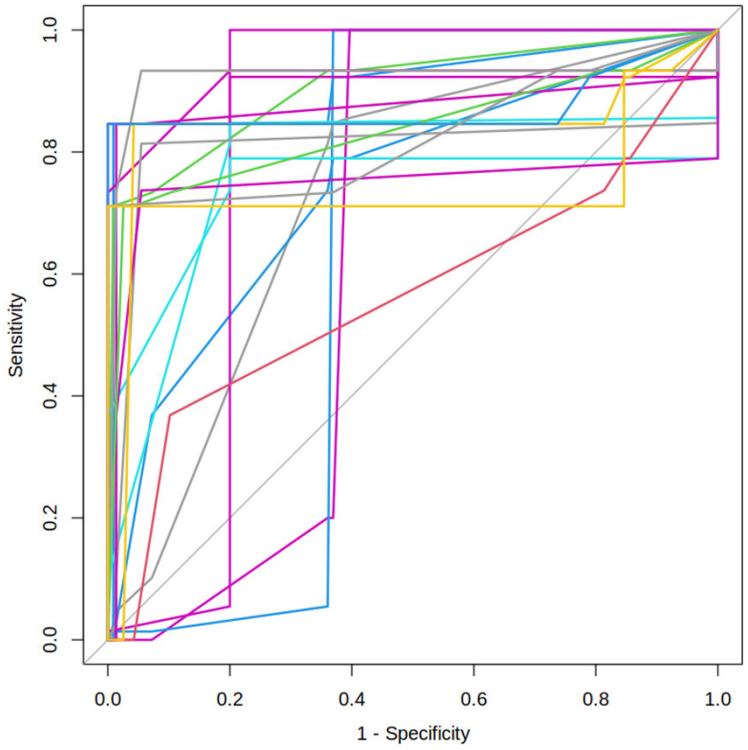
- Perform 10-fold Cross-Validation to find out the best size and k

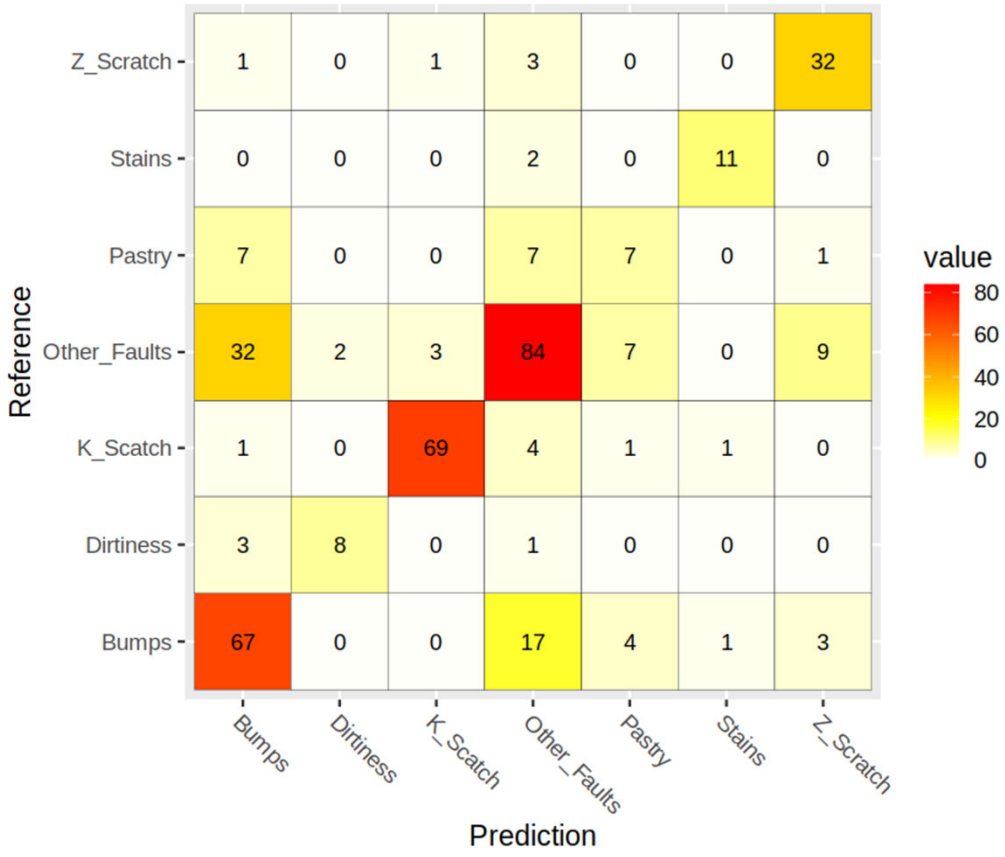
```
No pre-processing
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 1397, 1395, 1395, 1397, 1398, 1398, ...
Resampling results across tuning parameters:

  size  k   Accuracy   Kappa
   55    1  0.6702883  0.5805000
   55    6  0.6688773  0.5752882
   55   11  0.6662684  0.5721432
   82    1  0.6693742  0.5793639
   82    6  0.6804209  0.5905890
   82   11  0.6792960  0.5906694
  110    1  0.6927662  0.6078704
  110    6  0.6945780  0.6093202
  110   11  0.6902600  0.6034987

Accuracy was used to select the optimal model using the largest value
The final values used for the model were size = 110 and k = 6.
```



Learning Vector Quantization (without PCA)

- Accuracy on training set = 0.771
- Accuracy on test test = 0.715

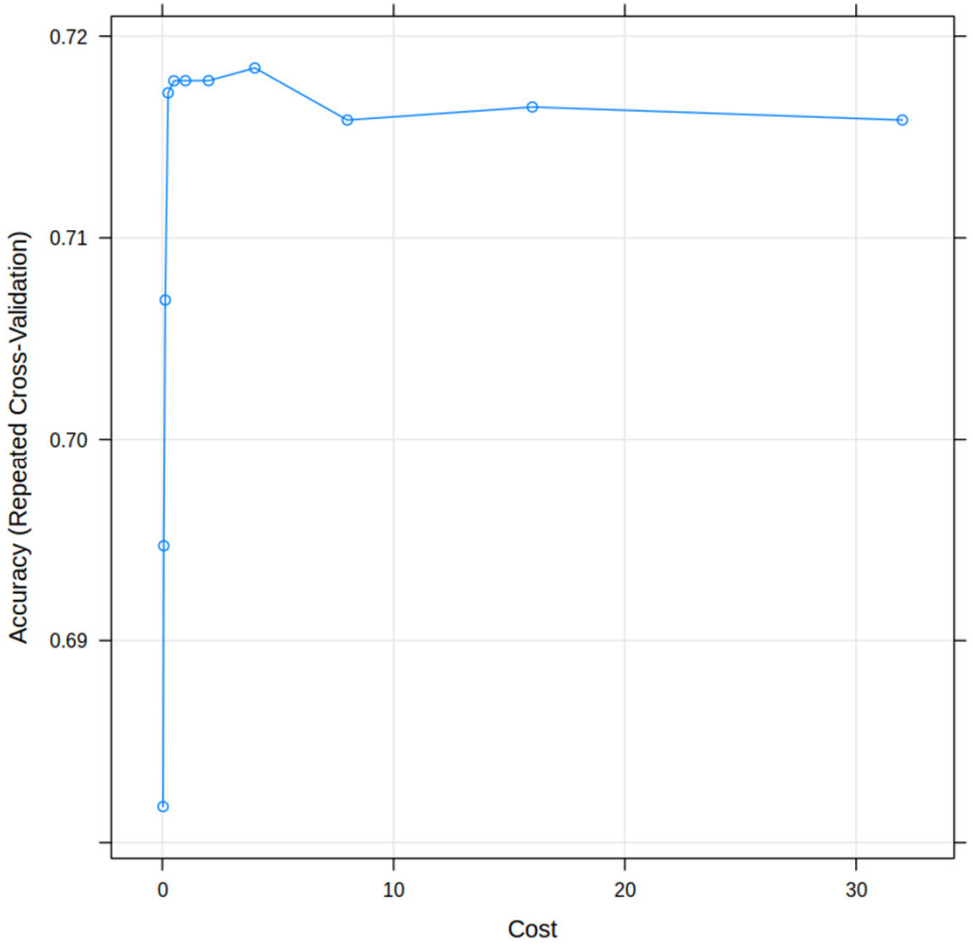




Support Vector Machine (with PCA, Linear Kernel)

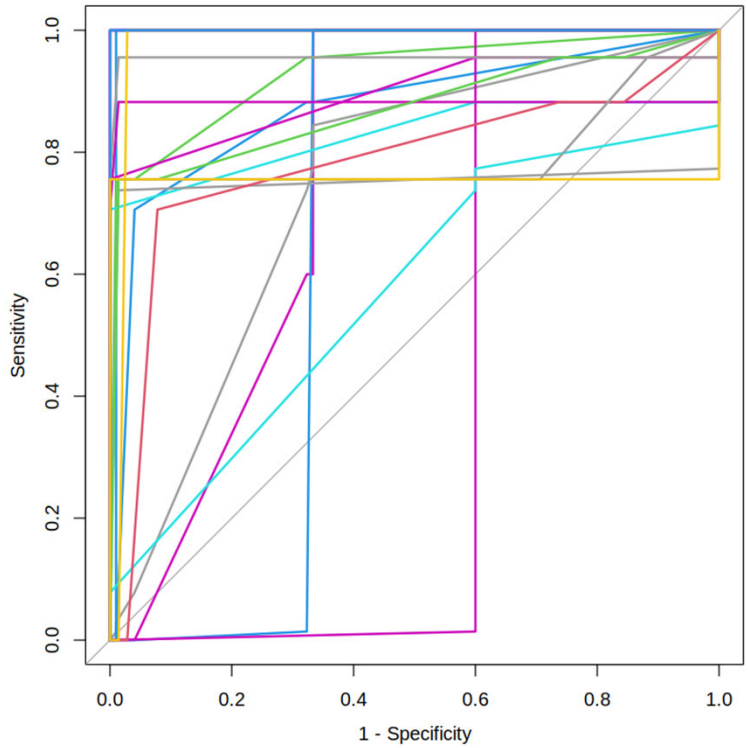
- Perform 10-fold Cross-Validation to find out the best C (C=4)

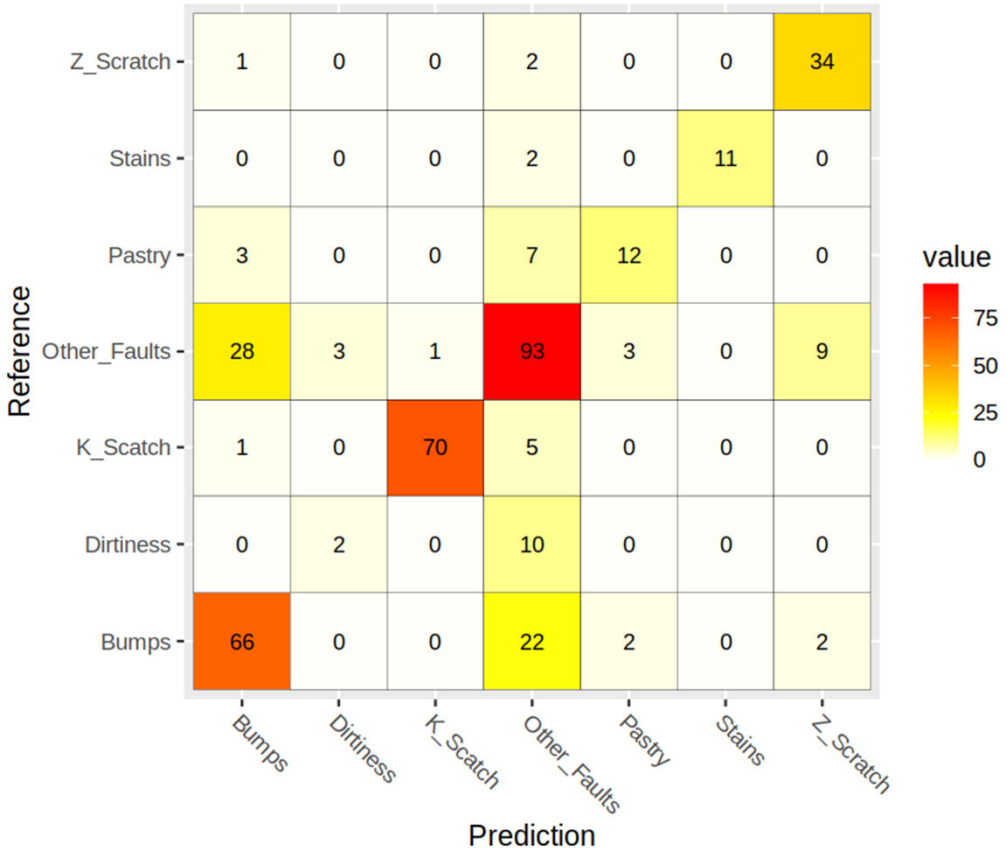
C	Accuracy	Kappa
0.03125	0.6817816	0.5780720
0.06250	0.6946894	0.6003648
0.12500	0.7069149	0.6183137
0.25000	0.7171880	0.6335957
0.50000	0.7177835	0.6356127
1.00000	0.7177918	0.6355241
2.00000	0.7177919	0.6353852
4.00000	0.7184205	0.6363781
8.00000	0.7158398	0.6326883
16.00000	0.7164851	0.6334521
32.00000	0.7158399	0.6326343



Support Vector Machine (with PCA, Linear Kernel)

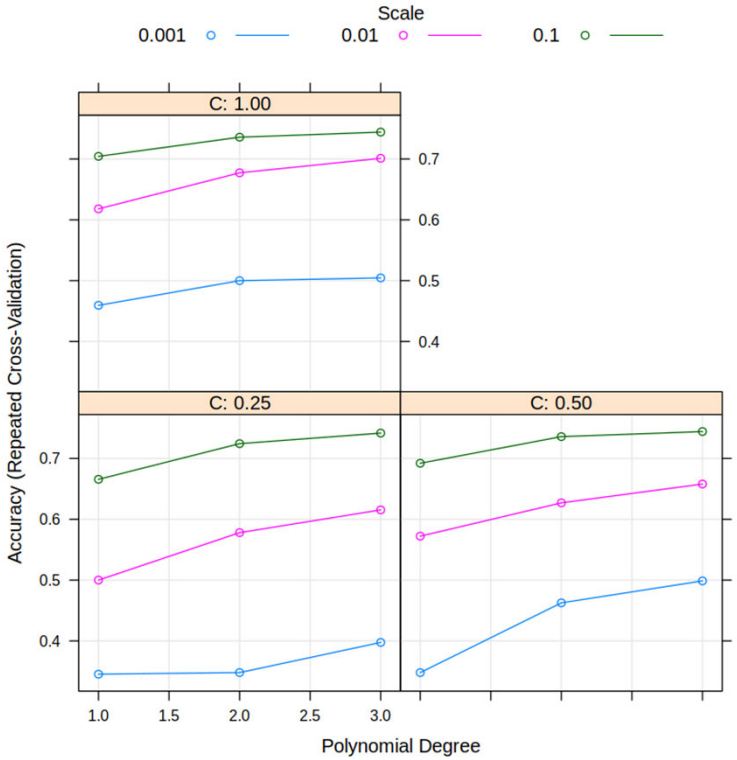
- Accuracy on training set = 0.747
- Accuracy on test test = 0.740





Support Vector Machine (with PCA, Polynomial Kernel)

- Perform 10-fold Cross-Validation to find out the best C, degree and scale

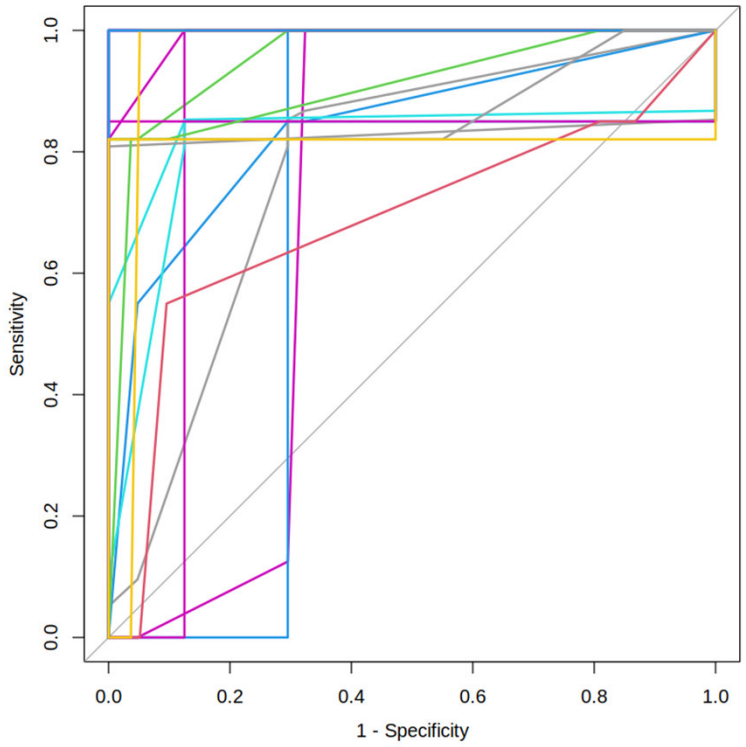


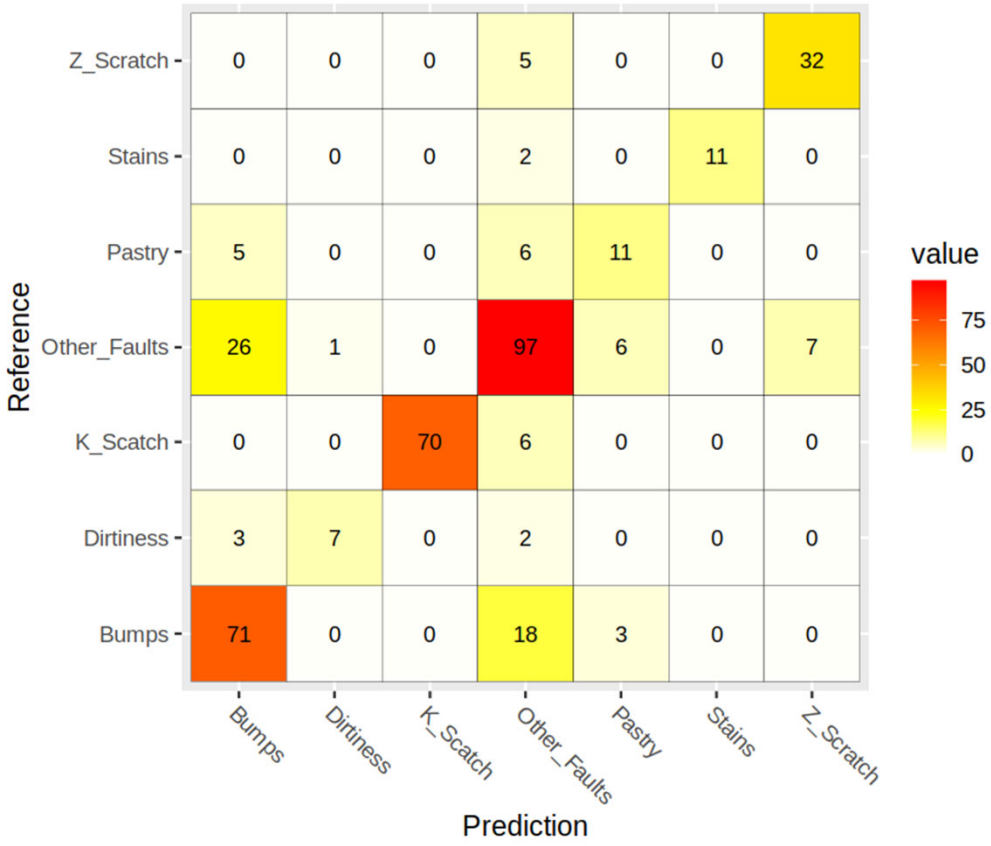
degree	scale	C	Accuracy	Kappa
1	0.001	0.25	0.3453699	0.00000000
1	0.001	0.50	0.3479590	0.00454193
1	0.001	1.00	0.4594358	0.19554283
1	0.010	0.25	0.5000083	0.26414889
1	0.010	0.50	0.5722156	0.39417243
1	0.010	1.00	0.6179945	0.47624769
1	0.100	0.25	0.6656895	0.55370872
1	0.100	0.50	0.6921005	0.59548656
1	0.100	1.00	0.7043549	0.61506735
2	0.001	0.25	0.3479590	0.00454193
2	0.001	0.50	0.4626328	0.20080824
2	0.001	1.00	0.5000083	0.26353720
2	0.010	0.25	0.5780593	0.40469127
2	0.010	0.50	0.6270024	0.48867175
2	0.010	1.00	0.6773069	0.56711092
2	0.100	0.25	0.7243349	0.64190688
2	0.100	0.50	0.7358700	0.65953800
2	0.100	1.00	0.7358613	0.66005084
3	0.001	0.25	0.3975638	0.09056534
3	0.001	0.50	0.4987180	0.26079904
3	0.001	1.00	0.5045164	0.27287763
3	0.010	0.25	0.6154259	0.46997484
3	0.010	0.50	0.6579511	0.53778503
3	0.010	1.00	0.7011413	0.60573303
3	0.100	0.25	0.7416765	0.66720308
3	0.100	0.50	0.7442117	0.67188130
3	0.100	1.00	0.7442077	0.67259818

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were degree = 3, scale = 0.1 and C = 0.5.

Support Vector Machine (with PCA, Polynomial Kernel)

- Accuracy on training set = 0.838
- Accuracy on test test = 0.769

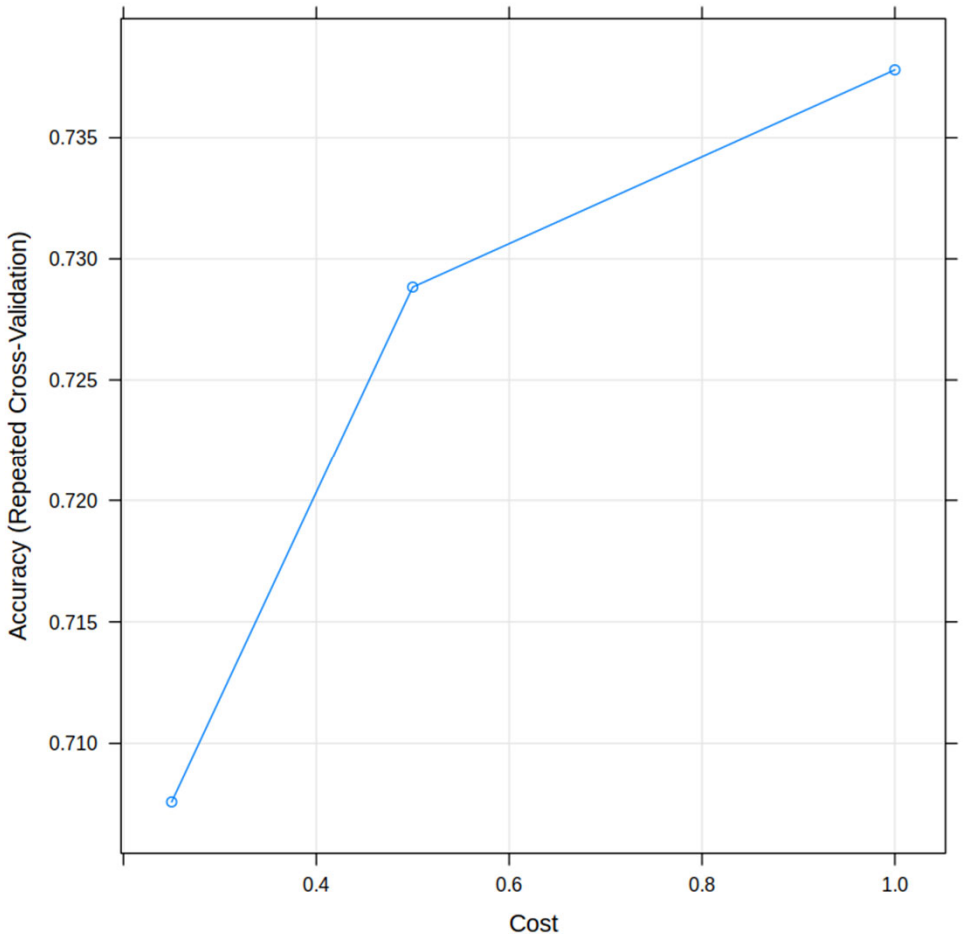




Support Vector Machine (with PCA, Radial Kernel)

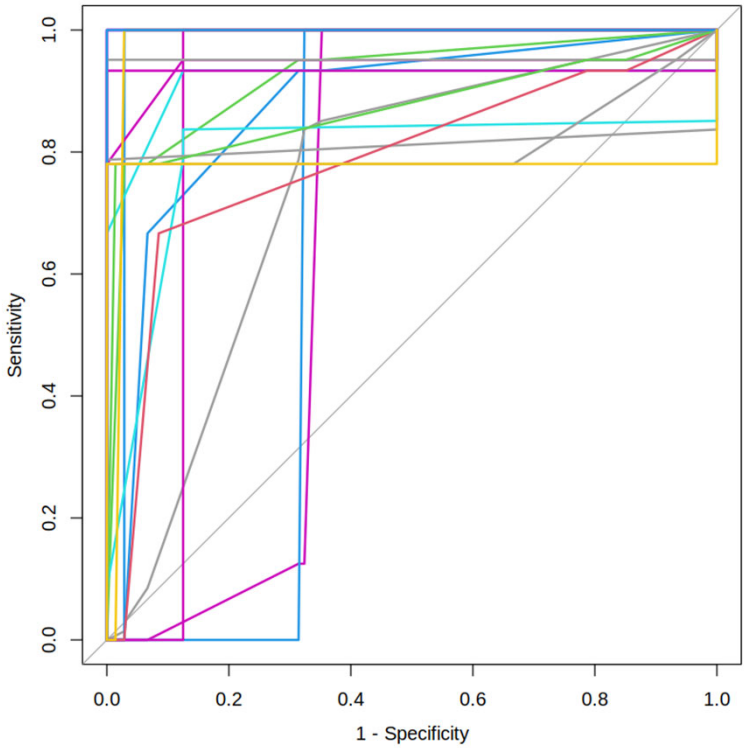
- Perform 10-fold Cross-Validation to find out the best C and sigma
- C = 1, sigma = 0.0647

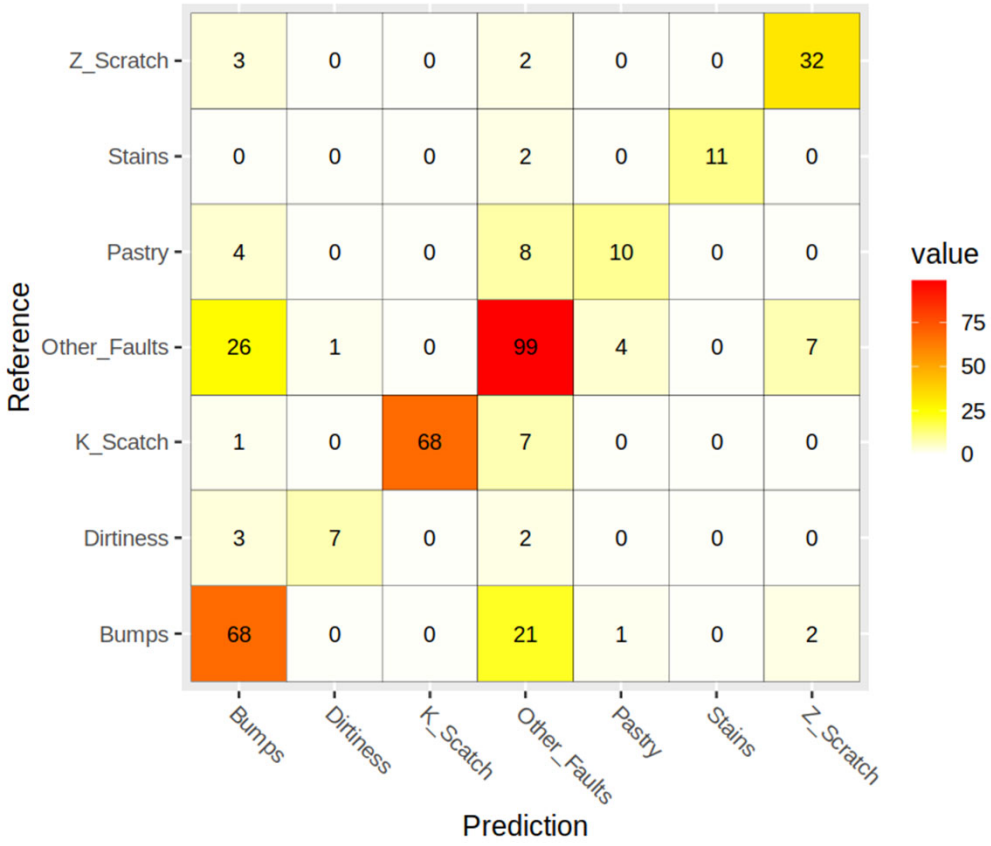
C	Accuracy	Kappa
0.25	0.7075936	0.6097245
0.50	0.7288353	0.6433275
1.00	0.7378014	0.6587338



Support Vector Machine (with PCA, Radial Kernel)

- Accuracy on training set = 0.789
- Accuracy on test test = 0.758

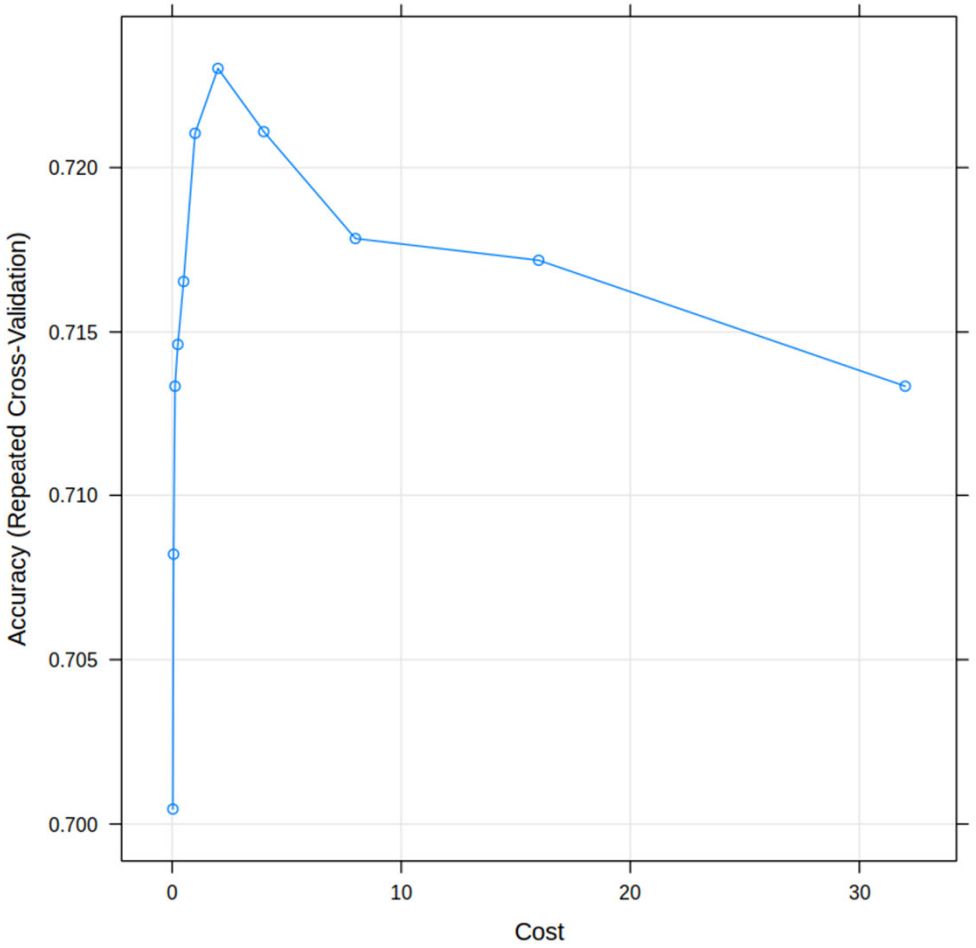




Support Vector Machine (without PCA, Linear Kernel)

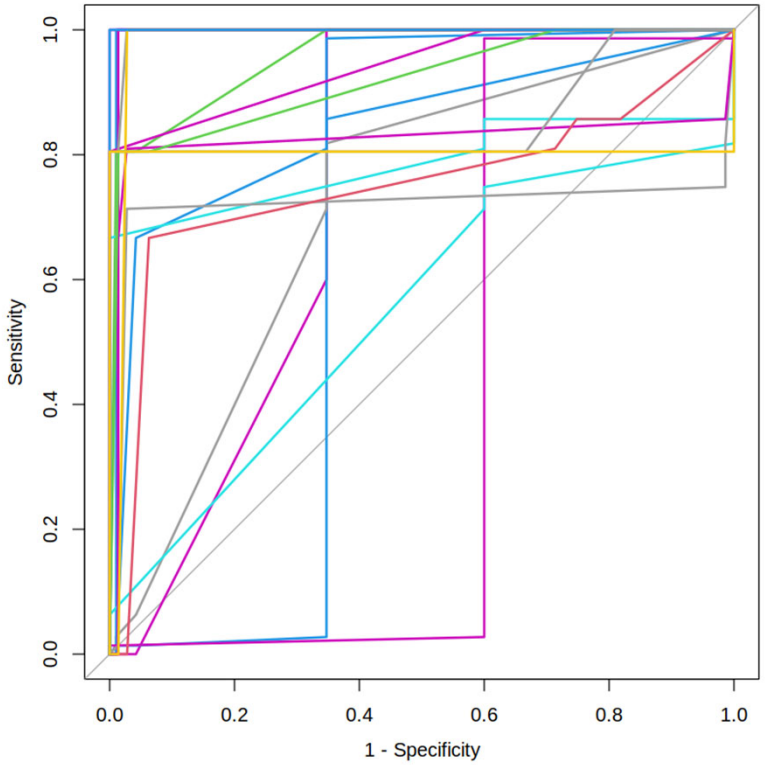
- Perform 10-fold Cross-Validation to find out the best C (C=2)

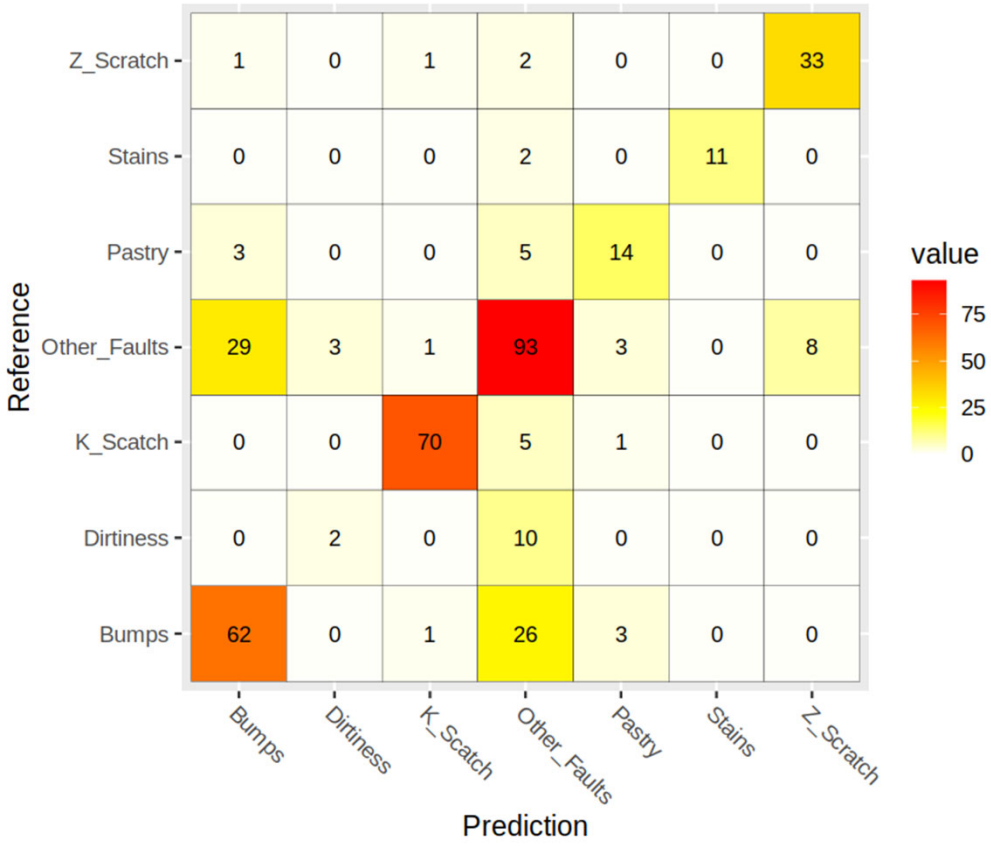
C	Accuracy	Kappa
0.03125	0.7004545	0.6066321
0.06250	0.7082090	0.6200125
0.12500	0.7133499	0.6291377
0.25000	0.7146196	0.6309865
0.50000	0.7165345	0.6343772
1.00000	0.7210424	0.6400224
2.00000	0.7230153	0.6422261
4.00000	0.7210921	0.6393130
8.00000	0.7178413	0.6354679
16.00000	0.7171799	0.6349848
32.00000	0.7133501	0.6308847



Support Vector Machine (without PCA, Linear Kernel)

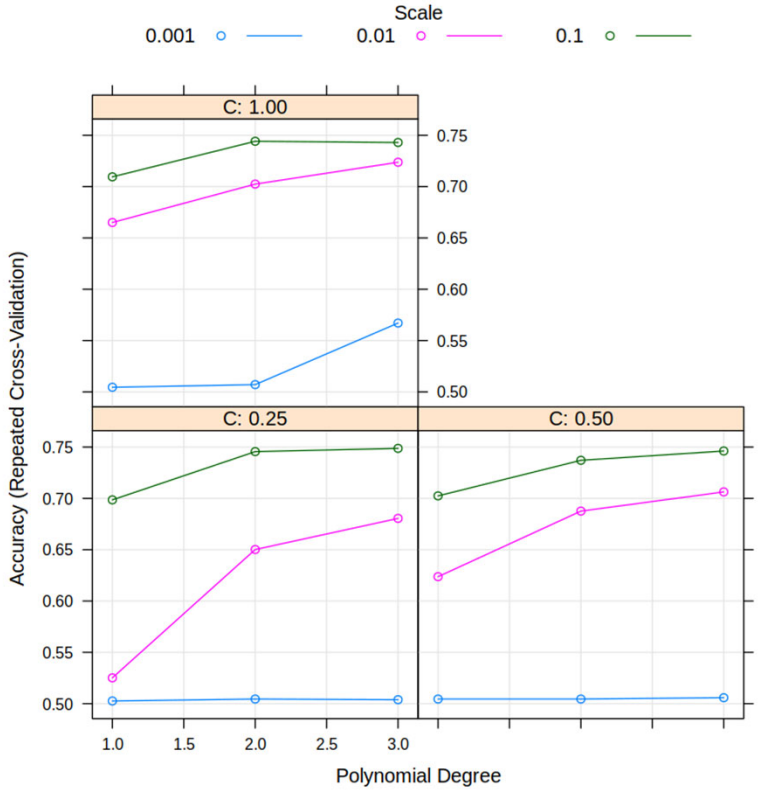
- Accuracy on training set = 0.747
- Accuracy on test test = 0.733





Support Vector Machine (without PCA, Polynomial Kernel)

- Perform 10-fold Cross-Validation to find out the best C, degree and scale

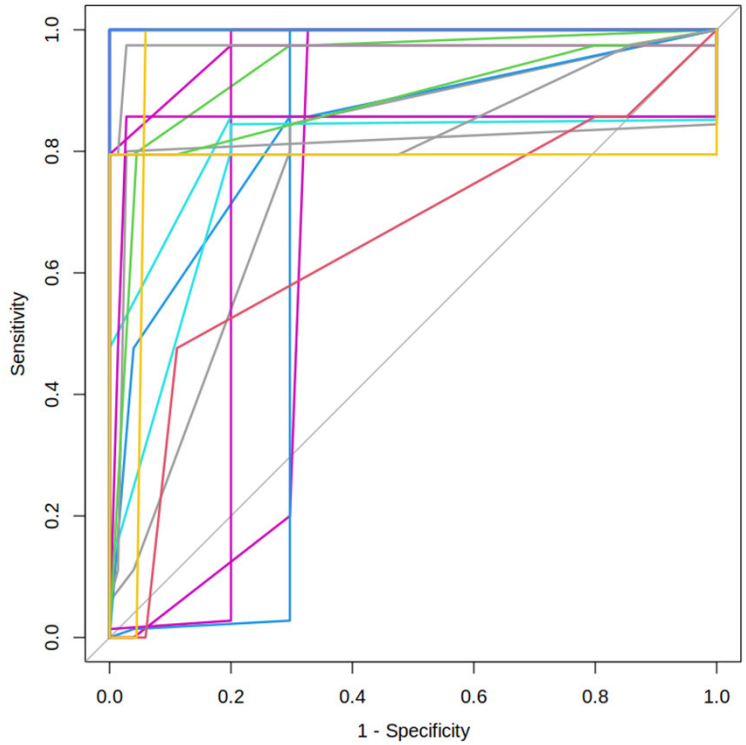


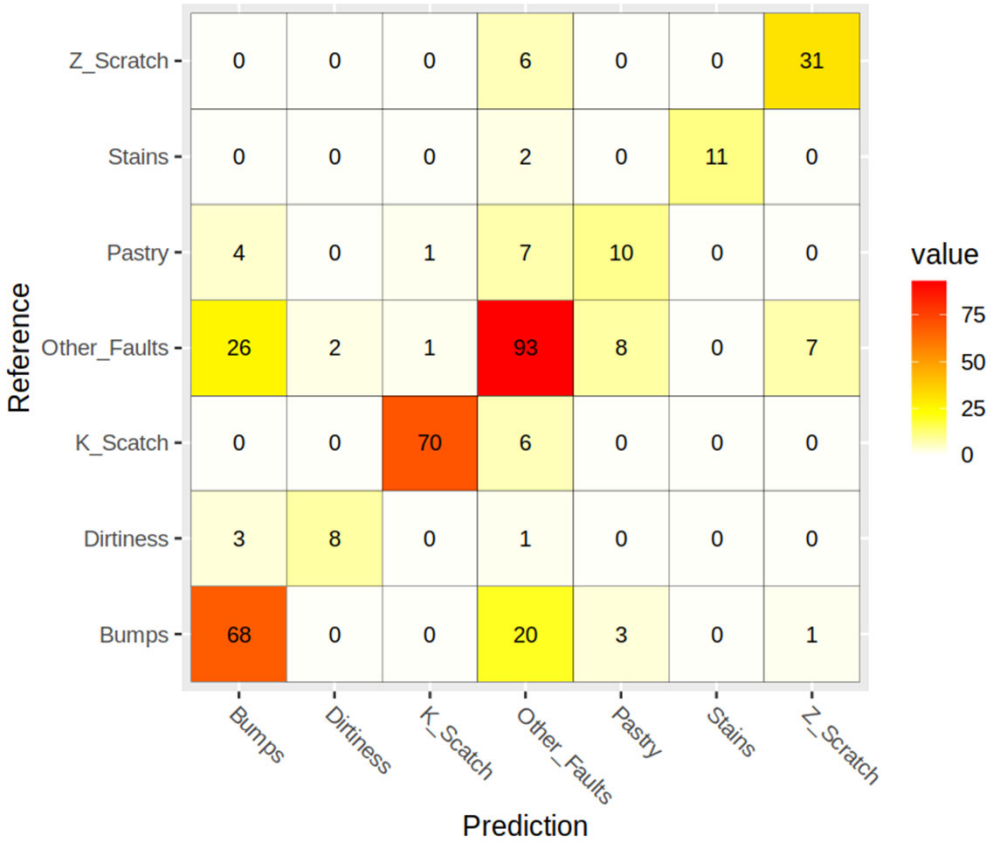
degree	scale	C	Accuracy	Kappa
1	0.001	0.25	0.5026099	0.2675620
1	0.001	0.50	0.5045082	0.2716578
1	0.001	1.00	0.5045082	0.2717562
1	0.010	0.25	0.5251585	0.3078909
1	0.010	0.50	0.6237188	0.4835367
1	0.010	1.00	0.6650192	0.5516766
1	0.100	0.25	0.6985479	0.6033523
1	0.100	0.50	0.7024232	0.6115400
1	0.100	1.00	0.7094913	0.6232100
2	0.001	0.25	0.5045082	0.2716578
2	0.001	0.50	0.5045082	0.2716531
2	0.001	1.00	0.5070890	0.2754839
2	0.010	0.25	0.6501676	0.5239752
2	0.010	0.50	0.6875965	0.5832345
2	0.010	1.00	0.7024152	0.6081644
2	0.100	0.25	0.7455516	0.6725086
2	0.100	0.50	0.7370858	0.6625354
2	0.100	1.00	0.7441581	0.6726394
3	0.001	0.25	0.5038588	0.2707999
3	0.001	0.50	0.5058069	0.2733639
3	0.001	1.00	0.5670585	0.3814207
3	0.010	0.25	0.6804955	0.5727194
3	0.010	0.50	0.7063025	0.6127933
3	0.010	1.00	0.7237186	0.6405888
3	0.100	0.25	0.7487033	0.6787525
3	0.100	0.50	0.7461558	0.6756885
3	0.100	1.00	0.7429507	0.6717666

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were degree = 3, scale = 0.1 and C = 0.25.

Support Vector Machine (without PCA, Polynomial Kernel)

- Accuracy on training set = 0.878
- Accuracy on test test = 0.748

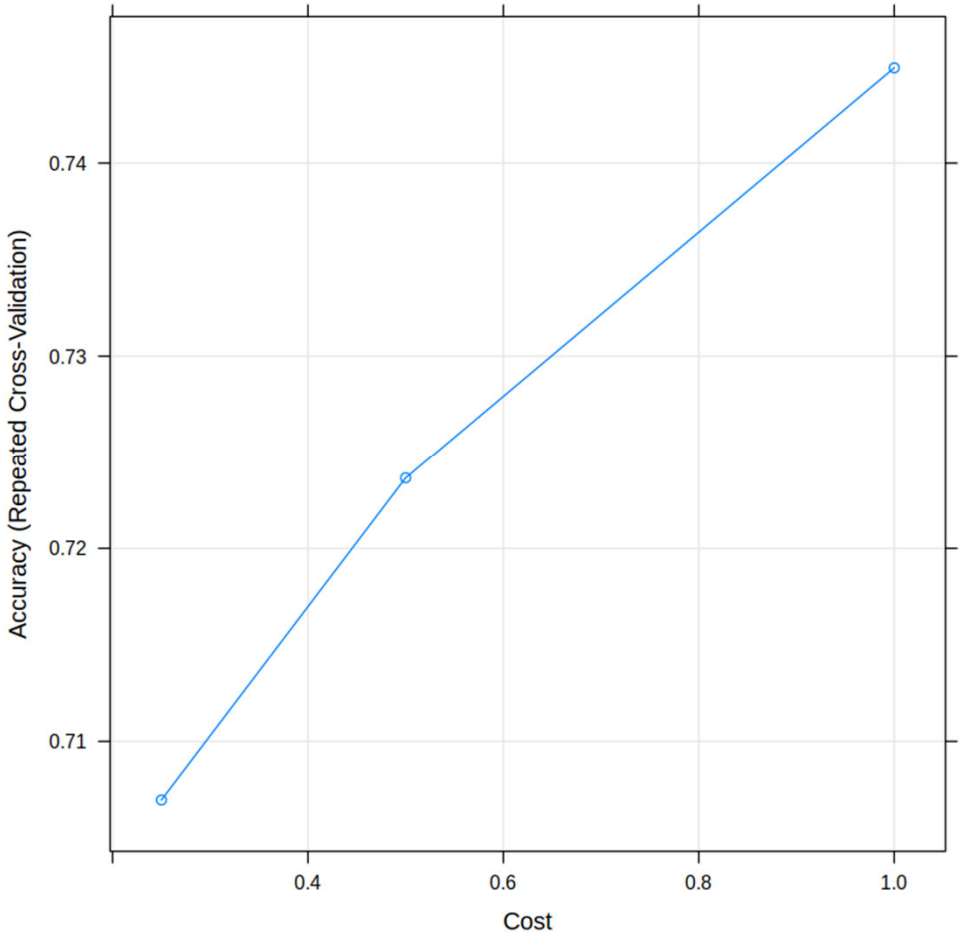




Support Vector Machine (without PCA, Radial Kernel)

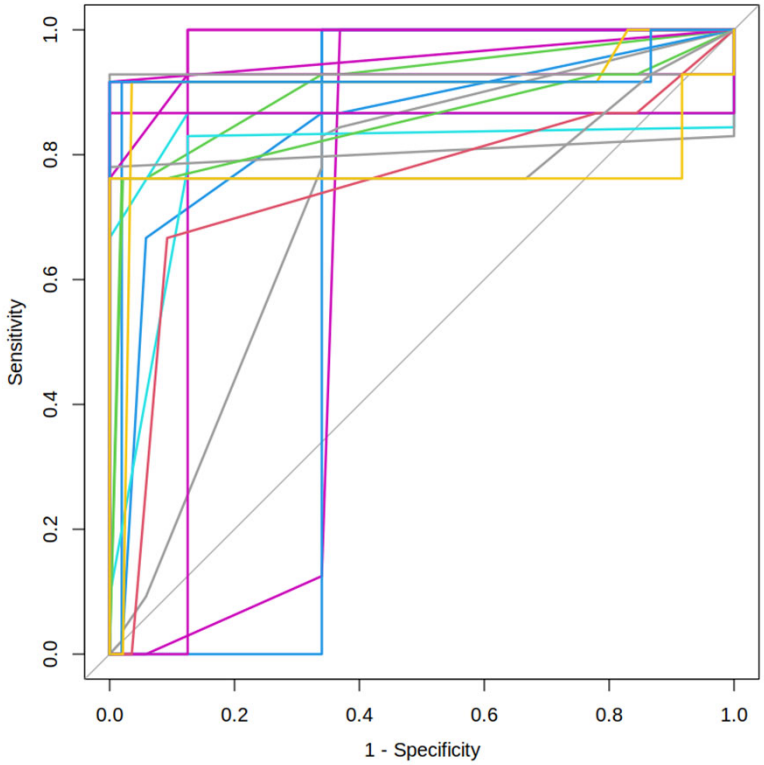
- Perform 10-fold Cross-Validation to find out the best C and sigma
- C = 1, sigma = 0.0338

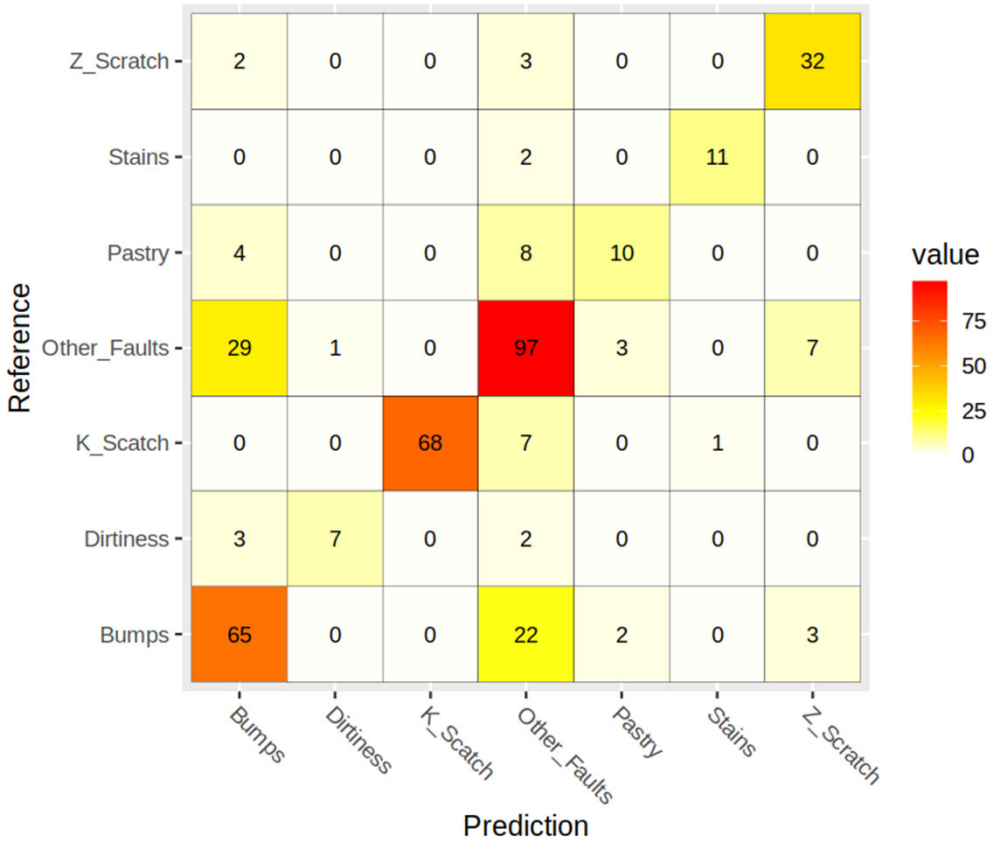
C	Accuracy	Kappa
0.25	0.7069652	0.6097298
0.50	0.7236611	0.6367780
1.00	0.7449312	0.6691088



Support Vector Machine (without PCA, Radial Kernel)

- Accuracy on training set = 0.798
- Accuracy on test test = 0.746





Conclusion

Training Set Accuracy

Model	Without PCA	With PCA
GNB	0.6101804	0.7003866
MLG	0.7403351	0.7197165
KNN	1	0.8228093
RF	1	1
BOOST	0.9574742	0.9329897
LVQ	0.7712629	0.6958763
SVM Linear	0.7474227	0.7474227
SVM Polynomial	0.8782216	0.8376289
SVM Radial	0.7976804	0.7893041
Best	KNN and RF	RF
Best overall	RF	

Testing Set Accuracy

Model	Without PCA	With PCA
GNB	0.6118252	0.6966581
MLG	0.7429306	0.722365
KNN	0.7429306	0.7429306
RF	0.781491	0.7660668
BOOST	0.7686375	0.7352185
LVQ	0.714653	0.6735219
SVM Linear	0.7326478	0.7403599
SVM Polynomial	0.748072	0.7686375
SVM Radial	0.7455013	0.7583548
Best	RF	SVM Polynomial
Best overall	RF without PCA	

Discussions

To improve accuracy, more advanced method can be used

Deep Learning methods

PCA may not always help.