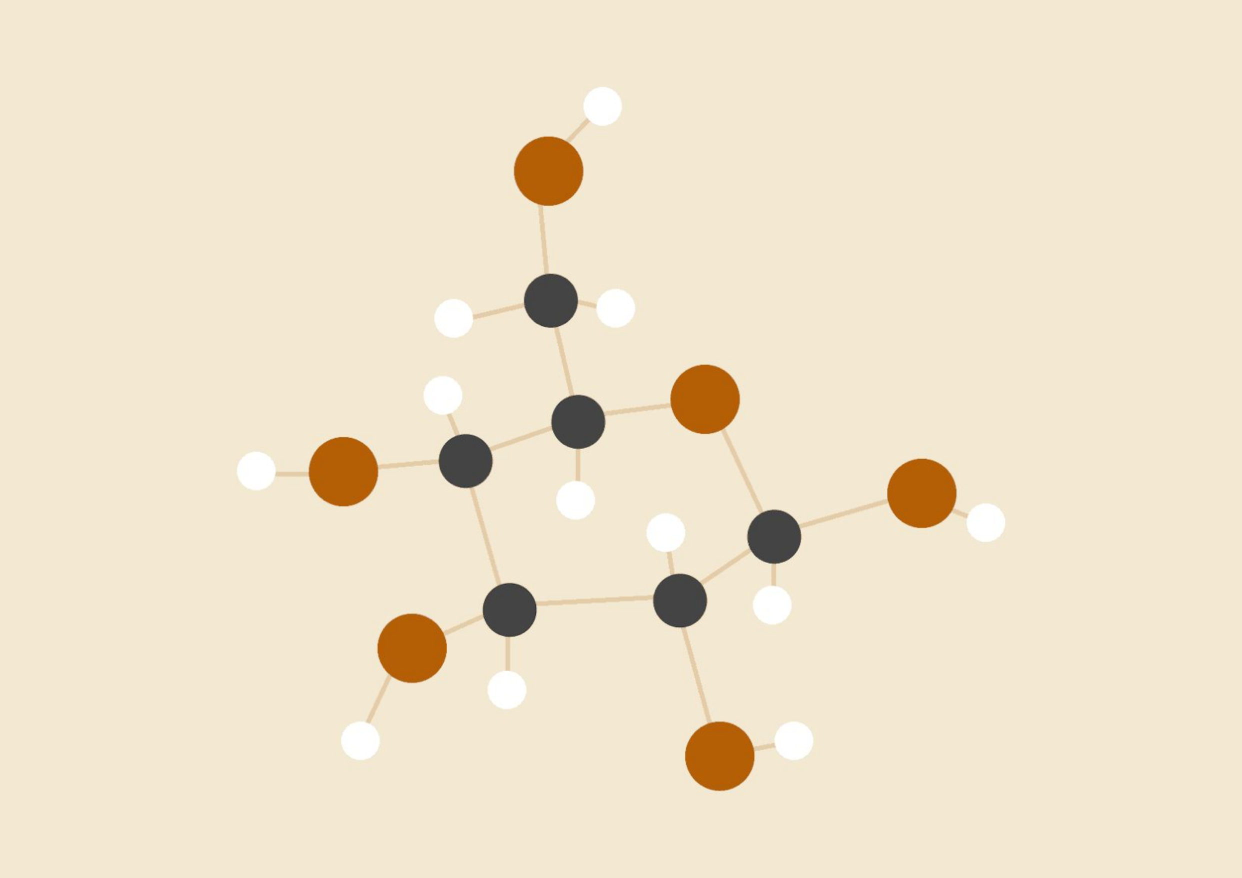
Project - I

Performance Analysis of Machine Learning and Pattern Recognition Algorithms for Malware Classification



Shreyas Raghunath

07.12.2020

INTRODUCTION

The aim of this project is to classify malware data by visualizing them as images and use machine learning approaches to classify them and also study and compare the accuracy by varying key factors that drive these approaches.

The approaches adopted for the following study include:

1. Neural Networks.
2. Principal Component Analysis - Using K nearest neighbours
3. Principal Component Analysis - Using Support vector machines

Methods

1. **Visualizing .bytes files as Images:**

   The main foundation block of this project is the curation of the dataset.

   The dataset provided contains 3 segments, the .asm files, the .bytesfiles and a csv sheet containing the output labels for corresponding.

   For the execution of this project the need was only for the .bytes files and the output labels.

   The steps of visualizing the .bytes files were implemented in the below fashion:

   - The output labels file was first imported with the list of file names and corresponding output labels
   - Using this order, the .bytes files were loaded and passed to the returnImage() function
   - The return image function performs the following actions:
     - Takes the entire.bytes file as a text bunch
     - Uses the split() function to consider all space separated text as one entity and store them
     - Since there is a binary component at the beginning of each line in the .bytes files, they are removed.
     - The occurrences of “??” which are special characters are noted down and these indices are initially assigned to 0(Since hex2dec() does not take negative values)

        - The values are passed to hex2dec() function to convert the hex pairs to decimal
        - The stored indices of “??” is now used to replace 0’s with -1’s
        - The results are then reshaped into 16 X (Total number of elements/16)
        - In cases where the number of elements is not divisible by 16 a few -1’s are added accordingly to make it divisible by 16
        - The obtained matrix is then resized into a 256 X 16 matrix.

    - The matrices obtained are then stored
    - The Stored matrices are then knit together by converting them as a row vector of 4096 elements and stacking them- the resultant matrix will be of the size 10868 X 4096.

Problems/Issues faced:

- *Assembling of the files based on the order in which they were listed in the .csv file*
    - Countered by using the id values from the csv file to import and store files

- *Dividing the values into rows and 16 columns as the number of elements varied*.
    - Countered by adding a not-fixed number of -1’s to the end to make the total number of elements divisible by 16

- Out of range values(lesser than -1) when images were resized.
    - Since the default method of resizing was bicubic, the end result of the image was out of range in a lot of cases
    - To counter this the bilinear method was used to resize the overall matrix.

Visualized images from the created dataset.

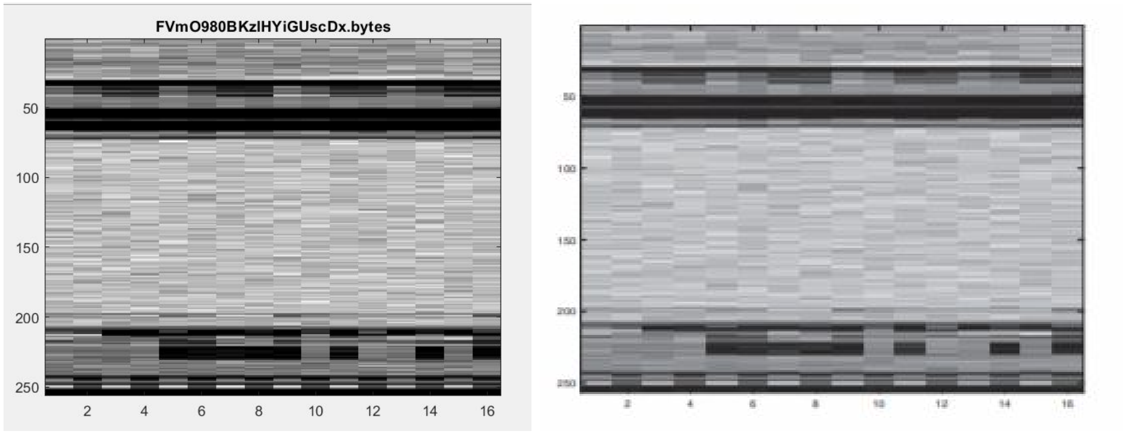


Image of ‘FVmO980BKzIHYiGUscDx.Bytes’ visualized as image and image of the same file from the reference paper

However the accuracy when this dataset was used was comparatively less, compared to the dataset provided with the project for reference.

So the other analysis has been done with the given reference dataset.

This issue might be due to the adding of -1’s to make the number of rows divisible by 16.

2. **Principal Component Analysis - Using K nearest neighbours:**

Large datasets are increasingly common and are often difficult to interpret.

Principal component analysis (PCA) is a technique for reducing the dimensionality of such datasets, increasing interpretability but at the same time minimizing information loss. It does so by creating new uncorrelated variables that successively maximize variance[1].

For this project principal component analysis is done using 2 methods of classification:

1. K Nearest neighbours
2. Support vector machines(Discussed in the next section)

The Process:

The initial process of principal component analysis remains the same for both of the above mentioned methods

- The data set is first imported and the input dataset is normalized, this process of normalization is to ensure that the values lie within a common scale/range.

- The normalized data is then passed into the PCA transformation function where the following takes place:
  - The covariance of the matrix is calculated using the COV() function
  - The obtained covariance result is then fed to the eig() function which returns the eigenvalues and eigenvectors.
  - The eigenvectors are then sorted in the descending order and the sorted indices are used to reorder the eigenvalues accordingly.
  - Based on the number of top N features fixed the matrix is shortened and a transform Matrix is calculated by multiplying the newly created feature matrix and the dataset
  -
  - The obtained matrix is then used for validation and the results are predicted.

- The project did mandate the above process to be applied using K-Fold validation to assert results.
- The K-fold validation is a resampling procedure used to evaluate machine learning models on a limited data sample.
  The general procedure is as follows:

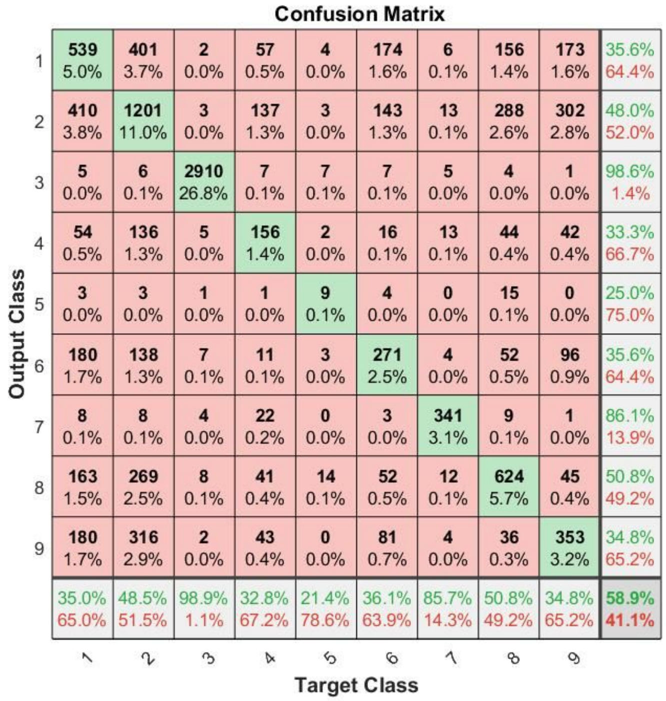
  1. Shuffle the dataset randomly(in or case it is a fixed set).
  2. Split the dataset into k groups
  3. For each unique group:
     i. Take the group as a hold out or test data set
     ii. Take the remaining groups as a training data set
     iii. Fit a model on the training set and evaluate it on the test set
  2. Retain the evaluation score and discard the model
  4. Summarize the skill of the model using the sample of model evaluation scores.

The Results

The results for PCA using knn were obtained by varying multiple aspects such as:

1. Varying the number of top features to be considered.
2. Varying the number of neighbours

Varying the number of top features to be considered:



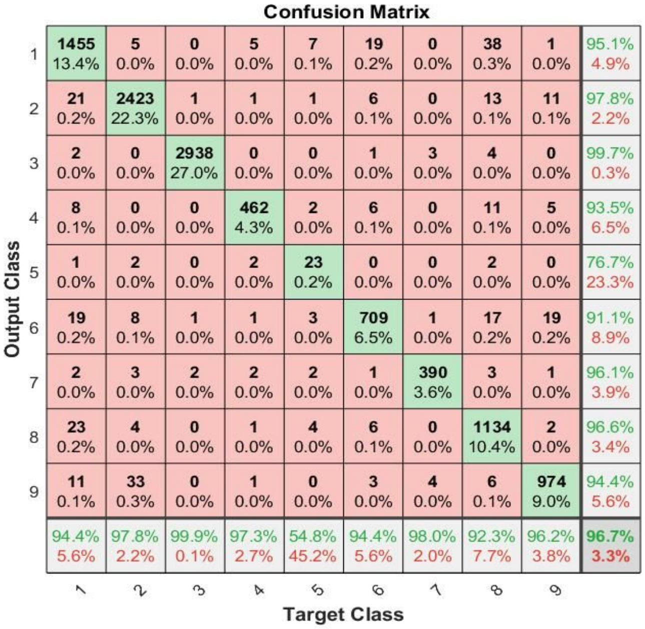
Confusion Matrix

Output Class	1	2	3	4	5	6	7	8	9	
1	539 5.0%	401 3.7%	2 0.0%	57 0.5%	4 0.0%	174 1.6%	6 0.1%	156 1.4%	173 1.6%	35.6% 64.4%
2	410 3.8%	1201 11.0%	3 0.0%	137 1.3%	3 0.0%	143 1.3%	13 0.1%	288 2.6%	302 2.8%	48.0% 52.0%
3	5 0.0%	6 0.1%	2910 26.8%	7 0.1%	7 0.1%	7 0.1%	5 0.0%	4 0.0%	1 0.0%	98.6% 1.4%
4	54 0.5%	136 1.3%	5 0.0%	156 1.4%	2 0.0%	16 0.1%	13 0.1%	44 0.4%	42 0.4%	33.3% 66.7%
5	3 0.0%	3 0.0%	1 0.0%	1 0.0%	9 0.1%	4 0.0%	0 0.0%	15 0.1%	0 0.0%	25.0% 75.0%
6	180 1.7%	138 1.3%	7 0.1%	11 0.1%	3 0.0%	271 2.5%	4 0.0%	52 0.5%	96 0.9%	35.6% 64.4%
7	8 0.1%	8 0.1%	4 0.0%	22 0.2%	0 0.0%	3 0.0%	341 3.1%	9 0.1%	1 0.0%	86.1% 13.9%
8	163 1.5%	269 2.5%	8 0.1%	41 0.4%	14 0.1%	52 0.5%	12 0.1%	624 5.7%	45 0.4%	50.8% 49.2%
9	180 1.7%	316 2.9%	2 0.0%	43 0.4%	0 0.0%	81 0.7%	4 0.0%	36 0.3%	353 3.2%	34.8% 65.2%
	35.0% 65.0%	48.5% 51.5%	98.9% 1.1%	32.8% 67.2%	21.4% 78.6%	36.1% 63.9%	85.7% 14.3%	50.8% 49.2%	34.8% 65.2%	58.9% 41.1%

Target Class

The above confusion matrix is when the **Number of features is 1 and 1 neighbour.**

The low accuracy can be attributed to the extremely low number of features being used. This means that 2 different classes with similar first feature could be grouped into 1 class.


Confusion Matrix
Output Class
Target Class

	1	2	3	4	5	6	7	8	9	
1	1455 13.4%	5 0.0%	0 0.0%	5 0.0%	7 0.1%	19 0.2%	0 0.0%	38 0.3%	1 0.0%	95.1% 4.9%
2	21 0.2%	2423 22.3%	1 0.0%	1 0.0%	1 0.0%	6 0.1%	0 0.0%	13 0.1%	11 0.1%	97.8% 2.2%
3	2 0.0%	0 0.0%	2938 27.0%	0 0.0%	0 0.0%	1 0.0%	3 0.0%	4 0.0%	0 0.0%	99.7% 0.3%
4	8 0.1%	0 0.0%	0 0.0%	462 4.3%	2 0.0%	6 0.1%	0 0.0%	11 0.1%	5 0.0%	93.5% 6.5%
5	1 0.0%	2 0.0%	0 0.0%	2 0.0%	23 0.2%	0 0.0%	0 0.0%	2 0.0%	0 0.0%	76.7% 23.3%
6	19 0.2%	8 0.1%	1 0.0%	1 0.0%	3 0.0%	709 6.5%	1 0.0%	17 0.2%	19 0.2%	91.1% 8.9%
7	2 0.0%	3 0.0%	2 0.0%	2 0.0%	2 0.0%	1 0.0%	390 3.6%	3 0.0%	1 0.0%	96.1% 3.9%
8	23 0.2%	4 0.0%	0 0.0%	1 0.0%	4 0.0%	6 0.1%	0 0.0%	1134 10.4%	2 0.0%	96.6% 3.4%
9	11 0.1%	33 0.3%	0 0.0%	1 0.0%	0 0.0%	3 0.0%	4 0.0%	6 0.1%	974 9.0%	94.4% 5.6%
	94.4% 5.6%	97.8% 2.2%	99.9% 0.1%	97.3% 2.7%	54.8% 45.2%	94.4% 5.6%	98.0% 2.0%	92.3% 7.7%	96.2% 3.8%	96.7% 3.3%

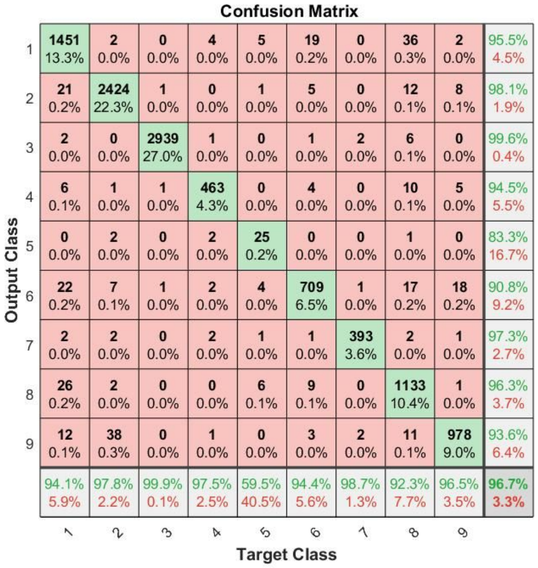
The above confusion matrix is when the **Number of features is 24 and 1 neighbour.**

The number features are far higher compared to the previous case but gives us a result which is very close to the expected result.

However the variance is very less beyond a point and becomes negligible.

Since the number of features are more, the computation time will also be equally higher.

We can try reducing the number features as prescribed in the reference paper to check if they coincide.



Confusion Matrix

Output Class	1	2	3	4	5	6	7	8	9	
1	1451 13.3%	2 0.0%	0 0.0%	4 0.0%	5 0.0%	19 0.2%	0 0.0%	36 0.3%	2 0.0%	95.5% 4.5%
2	21 0.2%	2424 22.3%	1 0.0%	0 0.0%	1 0.0%	5 0.0%	0 0.0%	12 0.1%	8 0.1%	98.1% 1.9%
3	2 0.0%	0 0.0%	2939 27.0%	1 0.0%	0 0.0%	1 0.0%	2 0.0%	6 0.1%	0 0.0%	99.6% 0.4%
4	6 0.1%	1 0.0%	1 0.0%	463 4.3%	0 0.0%	4 0.0%	0 0.0%	10 0.1%	5 0.0%	94.5% 5.5%
5	0 0.0%	2 0.0%	0 0.0%	2 0.0%	25 0.2%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	83.3% 16.7%
6	22 0.2%	7 0.1%	1 0.0%	2 0.0%	4 0.0%	709 6.5%	1 0.0%	17 0.2%	18 0.2%	90.8% 9.2%
7	2 0.0%	2 0.0%	0 0.0%	2 0.0%	1 0.0%	1 0.0%	393 3.6%	2 0.0%	1 0.0%	97.3% 2.7%
8	26 0.2%	2 0.0%	0 0.0%	0 0.0%	6 0.1%	9 0.1%	0 0.0%	1133 10.4%	1 0.0%	96.3% 3.7%
9	12 0.1%	38 0.3%	0 0.0%	1 0.0%	0 0.0%	3 0.0%	2 0.0%	11 0.1%	978 9.0%	93.6% 6.4%
	94.1% 5.9%	97.8% 2.2%	99.9% 0.1%	97.5% 2.5%	59.5% 40.5%	94.4% 5.6%	98.7% 1.3%	92.3% 7.7%	96.5% 3.5%	96.7% 3.3%

Target Class

The above confusion matrix is when the **Number of features is 12 and 1 neighbour.**

Since we have very similar results for both 12 and 24 features we can stick to 12 features.

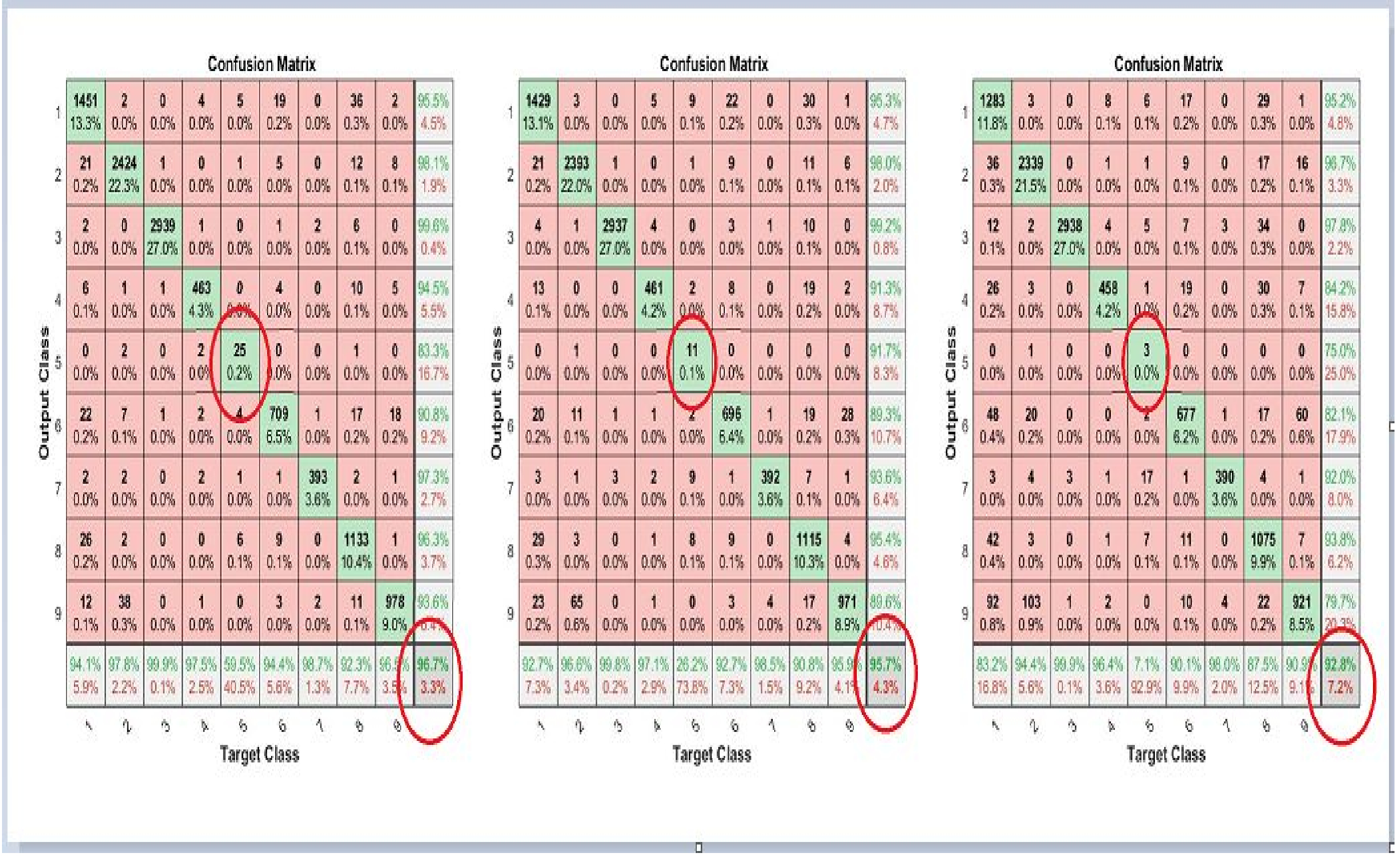
More over the number of correct results for SIMDA(Class V) which has the lowest number of samples has better accuracy when we use 12 features.

Method	Number of Features	Final Accuracy
PCA - KNN	1	58.9
PCA - KNN	24	96.7
PCA - KNN	12	96.7

Varying the number of Neighbours:

In this method we vary the number of neighbours used for Knn.

Since we know that the apt number of features is 12 from the first variation of feature we will stick to having 12 top features for variation of all neighbours.



Confusion Matrix

Output Class	1	2	3	4	5	6	7	8	9	
1	1451 13.3%	2 0.0%	0 0.0%	4 0.0%	5 0.0%	19 0.2%	0 0.0%	36 0.3%	2 0.0%	95.5% 4.5%
2	21 0.2%	2424 22.3%	1 0.0%	0 0.0%	1 0.0%	5 0.0%	0 0.0%	12 0.1%	8 0.1%	98.1% 1.9%
3	2 0.0%	0 0.0%	2939 27.0%	1 0.0%	0 0.0%	1 0.0%	2 0.0%	6 0.1%	0 0.0%	99.6% 0.4%
4	6 0.1%	1 0.0%	1 0.0%	463 4.3%	0 0.0%	4 0.0%	0 0.0%	10 0.1%	5 0.0%	94.5% 5.5%
5	0 0.0%	2 0.0%	0 0.0%	2 0.0%	25 0.2%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	83.3% 16.7%
6	22 0.2%	7 0.1%	1 0.0%	2 0.0%	4 0.0%	709 6.5%	1 0.0%	17 0.2%	18 0.2%	90.8% 9.2%
7	2 0.0%	2 0.0%	0 0.0%	2 0.0%	1 0.0%	1 0.0%	393 3.6%	2 0.0%	1 0.0%	97.3% 2.7%
8	26 0.2%	2 0.0%	0 0.0%	0 0.0%	6 0.1%	9 0.1%	0 0.0%	1133 10.4%	1 0.0%	96.3% 3.7%
9	12 0.1%	38 0.3%	0 0.0%	1 0.0%	0 0.0%	3 0.0%	2 0.0%	11 0.1%	978 9.0%	93.6% 6.4%
	94.1% 5.9%	97.8% 2.2%	99.9% 0.1%	97.5% 2.5%	59.5% 40.5%	94.4% 5.6%	98.7% 1.3%	92.3% 7.7%	96.5% 3.5%	96.7% 3.3%

Target Class

Confusion Matrix

Output Class	1	2	3	4	5	6	7	8	9	
1	1429 13.1%	3 0.0%	0 0.0%	5 0.0%	9 0.1%	22 0.2%	0 0.0%	30 0.3%	1 0.0%	95.3% 4.7%
2	21 0.2%	2393 22.0%	1 0.0%	0 0.0%	1 0.0%	9 0.1%	0 0.0%	11 0.1%	6 0.1%	98.0% 2.0%
3	4 0.0%	1 0.0%	2937 27.0%	4 0.0%	0 0.0%	3 0.0%	1 0.0%	10 0.1%	0 0.0%	99.2% 0.8%
4	13 0.1%	0 0.0%	0 0.0%	461 4.2%	2 0.0%	8 0.1%	0 0.0%	19 0.2%	2 0.0%	91.3% 8.7%
5	0 0.0%	1 0.0%	0 0.0%	0 0.0%	11 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	91.7% 8.3%
6	20 0.2%	11 0.1%	1 0.0%	1 0.0%	2 0.0%	696 6.4%	1 0.0%	19 0.2%	28 0.3%	89.3% 10.7%
7	3 0.0%	1 0.0%	3 0.0%	2 0.0%	9 0.1%	1 0.0%	392 3.6%	7 0.1%	1 0.0%	93.6% 6.4%
8	29 0.3%	3 0.0%	0 0.0%	1 0.0%	8 0.1%	9 0.1%	0 0.0%	1115 10.3%	4 0.0%	95.4% 4.6%
9	23 0.2%	65 0.6%	0 0.0%	1 0.0%	0 0.0%	3 0.0%	4 0.0%	17 0.2%	971 8.9%	89.6% 10.4%
	92.7% 7.3%	96.6% 3.4%	99.8% 0.2%	97.1% 2.9%	26.2% 73.8%	92.7% 7.3%	98.5% 1.5%	90.8% 9.2%	95.9% 4.1%	95.7% 4.3%

Target Class

Confusion Matrix

Output Class	1	2	3	4	5	6	7	8	9	
1	1283 11.8%	3 0.0%	0 0.0%	8 0.1%	6 0.1%	17 0.2%	0 0.0%	29 0.3%	1 0.0%	95.2% 4.8%
2	36 0.3%	2339 21.5%	0 0.0%	1 0.0%	1 0.0%	9 0.1%	0 0.0%	17 0.2%	16 0.1%	96.7% 3.3%
3	12 0.1%	2 0.0%	2938 27.0%	4 0.0%	5 0.0%	7 0.1%	3 0.0%	34 0.3%	0 0.0%	97.8% 2.2%
4	26 0.2%	3 0.0%	0 0.0%	458 4.2%	1 0.0%	19 0.2%	0 0.0%	30 0.3%	7 0.1%	84.2% 15.8%
5	0 0.0%	1 0.0%	0 0.0%	0 0.0%	3 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	75.0% 25.0%
6	48 0.4%	20 0.2%	0 0.0%	0 0.0%	2 0.0%	677 6.2%	1 0.0%	17 0.2%	60 0.6%	82.1% 17.9%
7	3 0.0%	4 0.0%	3 0.0%	1 0.0%	17 0.2%	1 0.0%	390 3.6%	4 0.0%	1 0.0%	92.0% 8.0%
8	42 0.4%	3 0.0%	0 0.0%	1 0.0%	7 0.1%	11 0.1%	0 0.0%	1075 9.9%	7 0.1%	93.8% 6.2%
9	92 0.8%	103 0.9%	1 0.0%	2 0.0%	0 0.0%	10 0.1%	4 0.0%	22 0.2%	921 8.5%	79.7% 20.3%
	83.2% 16.8%	94.4% 5.6%	99.9% 0.1%	96.4% 3.6%	7.1% 92.9%	90.1% 9.9%	98.0% 2.0%	87.5% 12.5%	90.9% 9.1%	92.8% 7.2%

Target Class

Fig 1: 12 Features - 1 Neighbour

Fig 2: 12 Features - 10 Neighbours

Fig 3: 12 Features - 100 Neighbours

A clear drop in the accuracy of the predictions can be found.

Especially if we closely notice the decrease in the values of class V(SIMDA) which has the lowest population in the dataset.

A clear understanding of Knn will help us in understanding this drop of behaviour more easily.

In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor[3].

As the values of K gets higher the clustering becomes more broad and the chances of focus on minute details are missed and hence a broader set of entities are classified under the wrong group which result in larger false positives and negatives for respective cases.

Method/Number of features	K - Nearest neighbours	Accuracy
PCA / 12	1	96.7
PCA / 12	10	96.7
PCA / 12	100	92.8

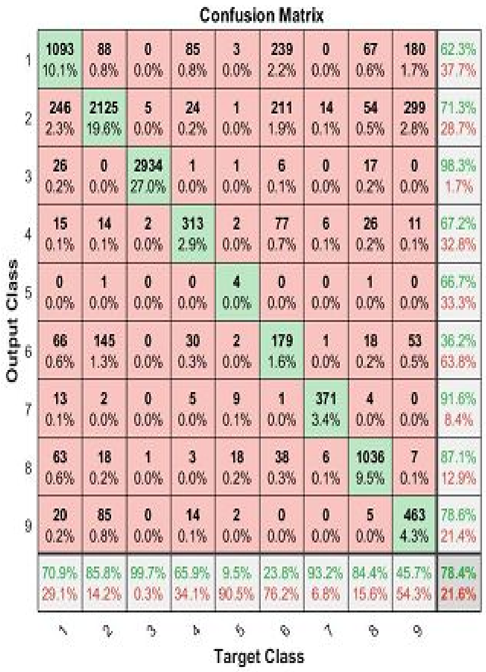
3. **Principal Component Analysis - Using Support vector machines:**

   Support-vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis[1].

   We use the same methodology for principal component analysis as earlier but use support vector machines to classify the entities.

   In this methodology we study the various kernels with which SVM's operate and tabulate the results

   1. Linear kernel

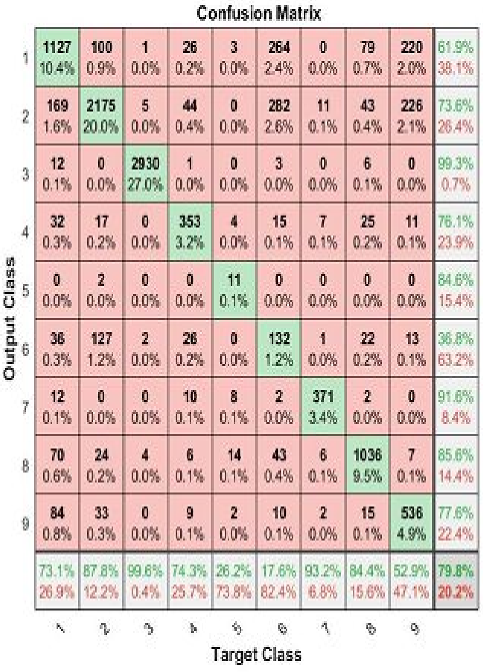


Confusion Matrix

Output Class

	1	2	3	4	5	6	7	8	9	
1	1093 10.1%	88 0.8%	0 0.0%	85 0.8%	3 0.0%	239 2.2%	0 0.0%	67 0.6%	180 1.7%	62.3% 37.7%
2	246 2.3%	2125 19.6%	5 0.0%	24 0.2%	1 0.0%	211 1.9%	14 0.1%	54 0.5%	299 2.8%	71.3% 28.7%
3	26 0.2%	0 0.0%	2934 27.0%	1 0.0%	1 0.0%	6 0.1%	0 0.0%	17 0.2%	0 0.0%	98.3% 1.7%
4	15 0.1%	14 0.1%	2 0.0%	313 2.9%	2 0.0%	77 0.7%	6 0.1%	26 0.2%	11 0.1%	67.2% 32.8%
5	0 0.0%	1 0.0%	0 0.0%	0 0.0%	4 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	66.7% 33.3%
6	66 0.6%	145 1.3%	0 0.0%	30 0.3%	2 0.0%	179 1.6%	1 0.0%	18 0.2%	53 0.5%	36.2% 63.8%
7	13 0.1%	2 0.0%	0 0.0%	5 0.0%	9 0.1%	1 0.0%	371 3.4%	4 0.0%	0 0.0%	91.6% 8.4%
8	63 0.6%	18 0.2%	1 0.0%	3 0.0%	18 0.2%	38 0.3%	6 0.1%	1036 9.5%	7 0.1%	87.1% 12.9%
9	20 0.2%	85 0.8%	0 0.0%	14 0.1%	2 0.0%	0 0.0%	0 0.0%	5 0.0%	463 4.3%	78.6% 21.4%
	70.9% 29.1%	85.8% 14.2%	99.7% 0.3%	65.9% 34.1%	9.5% 90.5%	23.8% 76.2%	93.2% 6.8%	84.4% 15.6%	45.7% 54.3%	78.4% 21.6%

Target Class

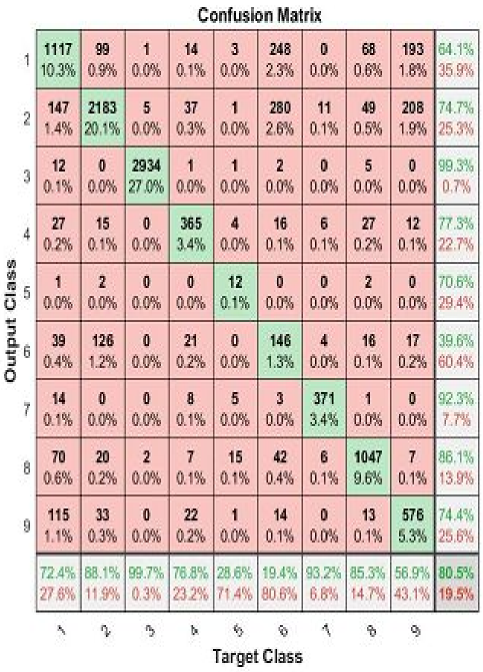


Confusion Matrix

Output Class

	1	2	3	4	5	6	7	8	9	
1	1127 10.4%	100 0.9%	1 0.0%	26 0.2%	3 0.0%	264 2.4%	0 0.0%	79 0.7%	220 2.0%	61.9% 38.1%
2	169 1.6%	2175 20.0%	5 0.0%	44 0.4%	0 0.0%	282 2.6%	11 0.1%	43 0.4%	226 2.1%	73.6% 26.4%
3	12 0.1%	0 0.0%	2930 27.0%	1 0.0%	0 0.0%	3 0.0%	0 0.0%	6 0.1%	0 0.0%	99.3% 0.7%
4	32 0.3%	17 0.2%	0 0.0%	353 3.2%	4 0.0%	15 0.1%	7 0.1%	25 0.2%	11 0.1%	76.1% 23.9%
5	0 0.0%	2 0.0%	0 0.0%	0 0.0%	11 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	84.6% 15.4%
6	36 0.3%	127 1.2%	2 0.0%	26 0.2%	0 0.0%	132 1.2%	1 0.0%	22 0.2%	13 0.1%	36.8% 63.2%
7	12 0.1%	0 0.0%	0 0.0%	10 0.1%	8 0.1%	2 0.0%	371 3.4%	2 0.0%	0 0.0%	91.6% 8.4%
8	70 0.6%	24 0.2%	4 0.0%	6 0.1%	14 0.1%	43 0.4%	6 0.1%	1036 9.5%	7 0.1%	85.6% 14.4%
9	84 0.8%	33 0.3%	0 0.0%	9 0.1%	2 0.0%	10 0.1%	2 0.0%	15 0.1%	536 4.9%	77.6% 22.4%
	73.1% 26.9%	87.8% 12.2%	99.6% 0.4%	74.3% 25.7%	26.2% 73.8%	17.6% 82.4%	93.2% 6.8%	84.4% 15.6%	52.9% 47.1%	79.8% 20.2%

Target Class



Confusion Matrix

Output Class

	1	2	3	4	5	6	7	8	9	
1	1117 10.3%	99 0.9%	1 0.0%	14 0.1%	3 0.0%	248 2.3%	0 0.0%	68 0.6%	193 1.8%	64.1% 35.9%
2	147 1.4%	2183 20.1%	5 0.0%	37 0.3%	1 0.0%	280 2.6%	11 0.1%	49 0.5%	208 1.9%	74.7% 25.3%
3	12 0.1%	0 0.0%	2934 27.0%	1 0.0%	1 0.0%	2 0.0%	0 0.0%	5 0.0%	0 0.0%	99.3% 0.7%
4	27 0.2%	15 0.1%	0 0.0%	365 3.4%	4 0.0%	16 0.1%	6 0.1%	27 0.2%	12 0.1%	77.3% 22.7%
5	1 0.0%	2 0.0%	0 0.0%	0 0.0%	12 0.1%	0 0.0%	0 0.0%	2 0.0%	0 0.0%	70.6% 29.4%
6	39 0.4%	126 1.2%	0 0.0%	21 0.2%	0 0.0%	146 1.3%	4 0.0%	16 0.1%	17 0.2%	39.6% 60.4%
7	14 0.1%	0 0.0%	0 0.0%	8 0.1%	5 0.0%	3 0.0%	371 3.4%	1 0.0%	0 0.0%	92.3% 7.7%
8	70 0.6%	20 0.2%	2 0.0%	7 0.1%	15 0.1%	42 0.4%	6 0.1%	1047 9.6%	7 0.1%	86.1% 13.9%
9	115 1.1%	33 0.3%	0 0.0%	22 0.2%	1 0.0%	14 0.1%	0 0.0%	13 0.1%	576 5.3%	74.4% 25.6%
	72.4% 27.6%	88.1% 11.9%	99.7% 0.3%	76.8% 23.2%	28.6% 71.4%	19.4% 80.6%	93.2% 6.8%	85.3% 14.7%	56.9% 43.1%	80.5% 19.5%

Target Class

Fig 1: 12 Features

Fig 2: 24 Features

Fig 3: 36 Features

As we can clearly see the linear kernel is not suited for this case.

This can be attributed to the lower number of features, and as the number of Features increases the accuracy increases.

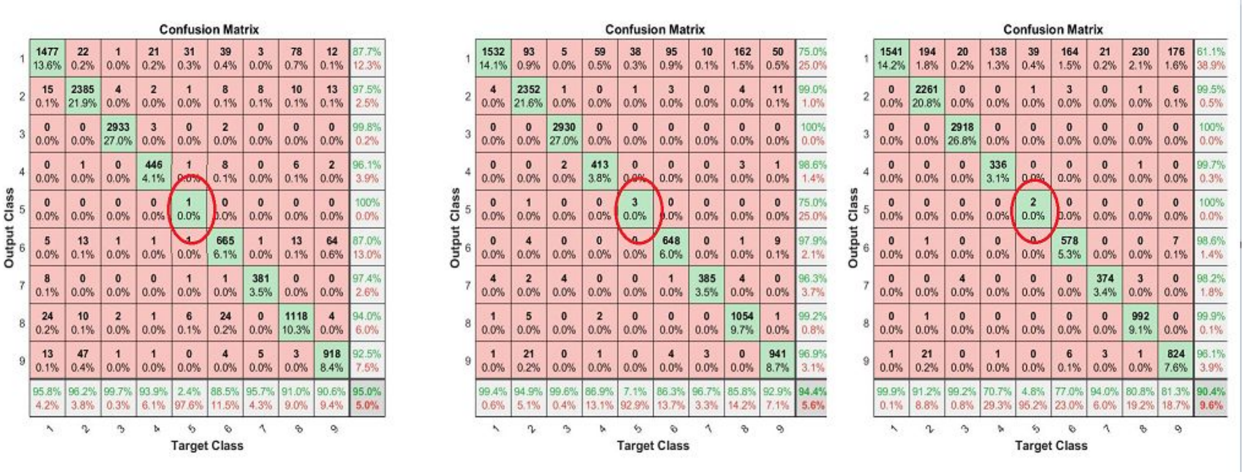
However a higher number of features results in higher computation so we can see if other kernels work well compared to linear and if they don't with lesser features we can adapt to a linear kernel.

2.Gaussian Kernel

The Gaussian kernel computed with a support vector is an exponentially decaying function in the input feature space, the maximum value of which is attained at the support vector and which decays uniformly in all directions around the support vector, leading to hyper-spherical contours of the kernel function.

The SVM classifier with the Gaussian kernel is simply a weighted linear combination of the kernel function computed between a data point and each of the support vectors.

Since the Gaussian kernel is recommender for number of features less than 25 we can check if the values are accurate.



Confusion Matrix

Output Class / Target Class

	1	2	3	4	5	6	7	8	9	
1	1477 13.6%	22 0.2%	1 0.0%	21 0.2%	31 0.3%	39 0.4%	3 0.0%	78 0.7%	12 0.1%	87.7% 12.3%
2	15 0.1%	2385 21.9%	4 0.0%	2 0.0%	1 0.0%	8 0.1%	8 0.1%	10 0.1%	13 0.1%	97.5% 2.5%
3	0 0.0%	0 0.0%	2933 27.0%	3 0.0%	0 0.0%	2 0.0%	0 0.0%	0 0.0%	0 0.0%	99.8% 0.2%
4	0 0.0%	1 0.0%	0 0.0%	446 4.1%	1 0.0%	8 0.1%	0 0.0%	6 0.1%	2 0.0%	96.1% 3.9%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
6	5 0.0%	13 0.1%	1 0.0%	1 0.0%	1 0.0%	665 6.1%	1 0.0%	13 0.1%	64 0.6%	87.0% 13.0%
7	8 0.1%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	1 0.0%	381 3.5%	0 0.0%	0 0.0%	97.4% 2.6%
8	24 0.2%	10 0.1%	2 0.0%	1 0.0%	6 0.1%	24 0.2%	0 0.0%	1118 10.3%	4 0.0%	94.0% 6.0%
9	13 0.1%	47 0.4%	1 0.0%	1 0.0%	0 0.0%	4 0.0%	5 0.0%	3 0.0%	918 8.4%	92.5% 7.5%
	95.8% 4.2%	96.2% 3.8%	99.7% 0.3%	93.9% 6.1%	2.4% 97.6%	88.5% 11.5%	95.7% 4.3%	91.0% 9.0%	90.6% 9.4%	95.0% 5.0%

Confusion Matrix

Output Class / Target Class

	1	2	3	4	5	6	7	8	9	
1	1532 14.1%	93 0.9%	5 0.0%	59 0.5%	38 0.3%	95 0.9%	10 0.1%	162 1.5%	50 0.5%	75.0% 25.0%
2	4 0.0%	2352 21.6%	1 0.0%	0 0.0%	1 0.0%	3 0.0%	0 0.0%	4 0.0%	11 0.1%	99.0% 1.0%
3	0 0.0%	0 0.0%	2930 27.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
4	0 0.0%	0 0.0%	2 0.0%	413 3.8%	0 0.0%	0 0.0%	0 0.0%	3 0.0%	1 0.0%	98.6% 1.4%
5	0 0.0%	1 0.0%	0 0.0%	0 0.0%	3 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	75.0% 25.0%
6	0 0.0%	4 0.0%	0 0.0%	0 0.0%	0 0.0%	648 6.0%	0 0.0%	1 0.0%	9 0.1%	97.9% 2.1%
7	4 0.0%	2 0.0%	4 0.0%	0 0.0%	0 0.0%	1 0.0%	385 3.5%	4 0.0%	0 0.0%	96.3% 3.7%
8	1 0.0%	5 0.0%	0 0.0%	2 0.0%	0 0.0%	0 0.0%	0 0.0%	1054 9.7%	1 0.0%	99.2% 0.8%
9	1 0.0%	21 0.2%	0 0.0%	1 0.0%	0 0.0%	4 0.0%	3 0.0%	0 0.0%	941 8.7%	96.9% 3.1%
	99.4% 0.6%	94.9% 5.1%	99.6% 0.4%	86.9% 13.1%	7.1% 92.9%	86.3% 13.7%	96.7% 3.3%	85.8% 14.2%	92.9% 7.1%	94.4% 5.6%

Confusion Matrix

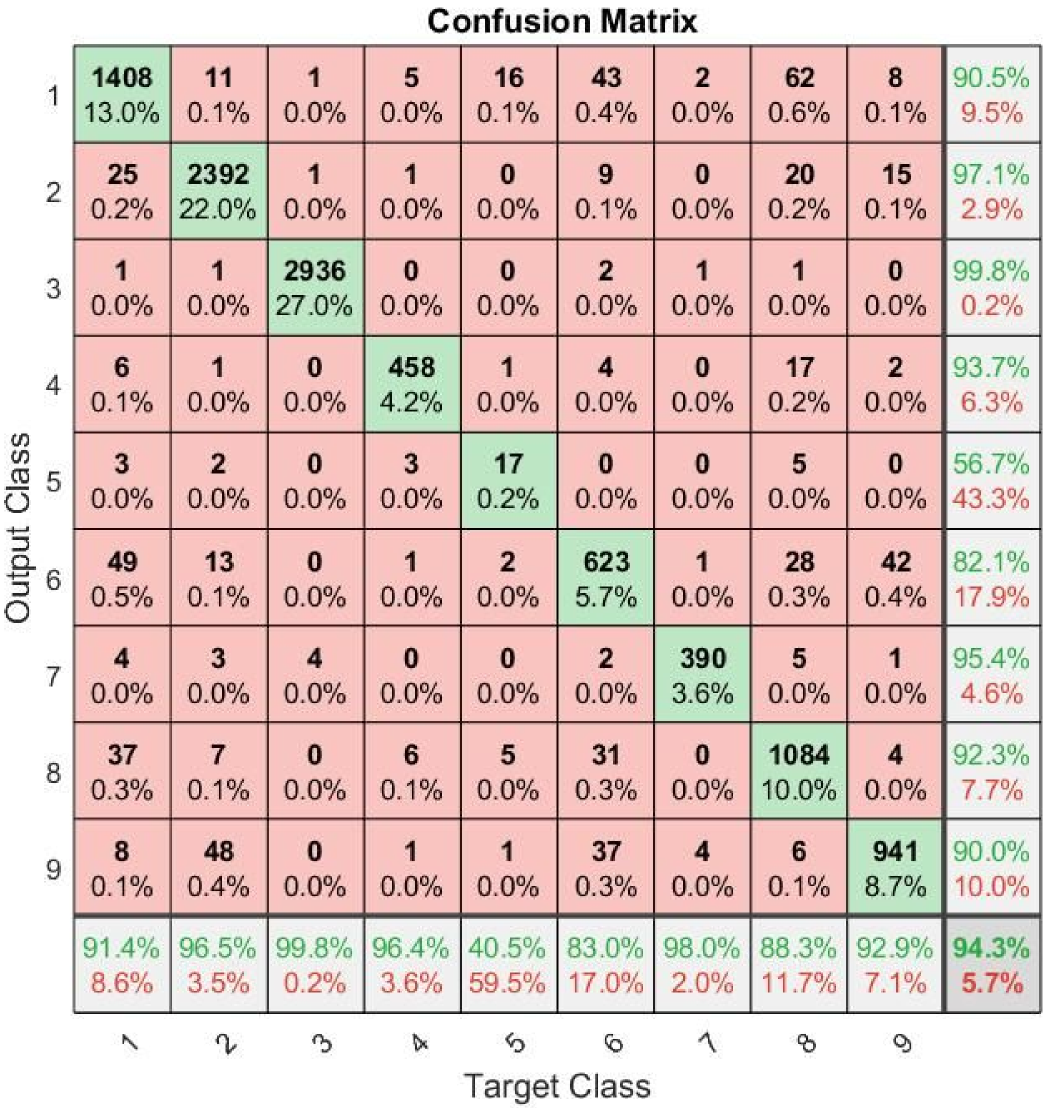
Output Class / Target Class

	1	2	3	4	5	6	7	8	9	
1	1541 14.2%	194 1.8%	20 0.2%	138 1.3%	39 0.4%	164 1.5%	21 0.2%	230 2.1%	176 1.6%	61.1% 38.9%
2	0 0.0%	2261 20.8%	0 0.0%	0 0.0%	1 0.0%	3 0.0%	0 0.0%	1 0.0%	6 0.1%	99.5% 0.5%
3	0 0.0%	0 0.0%	2918 26.8%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
4	0 0.0%	0 0.0%	0 0.0%	336 3.1%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	99.7% 0.3%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
6	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	578 5.3%	0 0.0%	0 0.0%	7 0.1%	98.6% 1.4%
7	0 0.0%	0 0.0%	4 0.0%	0 0.0%	0 0.0%	0 0.0%	374 3.4%	3 0.0%	0 0.0%	98.2% 1.8%
8	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	992 9.1%	0 0.0%	99.9% 0.1%
9	1 0.0%	21 0.2%	0 0.0%	1 0.0%	0 0.0%	6 0.1%	3 0.0%	1 0.0%	824 7.6%	96.1% 3.9%
	99.9% 0.1%	91.2% 8.8%	99.2% 0.8%	70.7% 29.3%	4.8% 95.2%	77.0% 23.0%	94.0% 6.0%	80.8% 19.2%	81.3% 18.7%	90.4% 9.6%

The 3 images represent gaussian kernel with 6, 12, 24 features respectively

The accuracy is pretty high in these cases however, as highlighted if you can see the prediction of the lowest available class is very very low in all these cases. So we can shortlist this but not finalize it as the accurate method.

3. Quadratic kernel



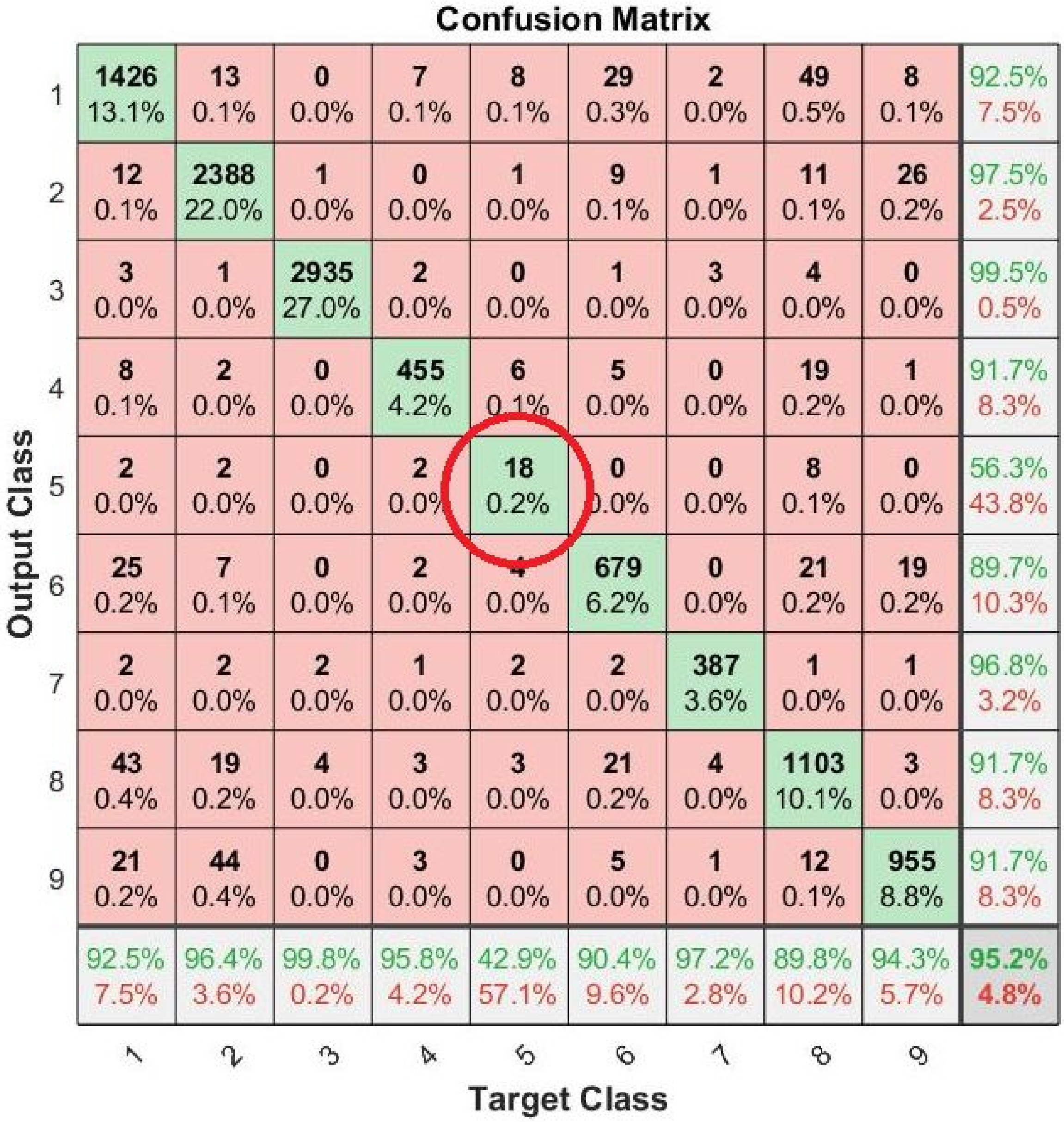
Confusion Matrix

Output Class \ Target Class	1	2	3	4	5	6	7	8	9	
1	1408 13.0%	11 0.1%	1 0.0%	5 0.0%	16 0.1%	43 0.4%	2 0.0%	62 0.6%	8 0.1%	90.5% 9.5%
2	25 0.2%	2392 22.0%	1 0.0%	1 0.0%	0 0.0%	9 0.1%	0 0.0%	20 0.2%	15 0.1%	97.1% 2.9%
3	1 0.0%	1 0.0%	2936 27.0%	0 0.0%	0 0.0%	2 0.0%	1 0.0%	1 0.0%	0 0.0%	99.8% 0.2%
4	6 0.1%	1 0.0%	0 0.0%	458 4.2%	1 0.0%	4 0.0%	0 0.0%	17 0.2%	2 0.0%	93.7% 6.3%
5	3 0.0%	2 0.0%	0 0.0%	3 0.0%	17 0.2%	0 0.0%	0 0.0%	5 0.0%	0 0.0%	56.7% 43.3%
6	49 0.5%	13 0.1%	0 0.0%	1 0.0%	2 0.0%	623 5.7%	1 0.0%	28 0.3%	42 0.4%	82.1% 17.9%
7	4 0.0%	3 0.0%	4 0.0%	0 0.0%	0 0.0%	2 0.0%	390 3.6%	5 0.0%	1 0.0%	95.4% 4.6%
8	37 0.3%	7 0.1%	0 0.0%	6 0.1%	5 0.0%	31 0.3%	0 0.0%	1084 10.0%	4 0.0%	92.3% 7.7%
9	8 0.1%	48 0.4%	0 0.0%	1 0.0%	1 0.0%	37 0.3%	4 0.0%	6 0.1%	941 8.7%	90.0% 10.0%
	91.4% 8.6%	96.5% 3.5%	99.8% 0.2%	96.4% 3.6%	40.5% 59.5%	83.0% 17.0%	98.0% 2.0%	88.3% 11.7%	92.9% 7.1%	94.3% 5.7%

The polynomial kernel looks not only at the given features of input samples to determine their similarity, but also combinations of these. In the context of regression analysis, such combinations are known as interaction features[4].

In this case the number however the accuracy is very similar to that of gaussian, it scores over gaussian in classifying class V more accurately where it score over gaussian kernel.

4. Polynomial Kernel



Confusion Matrix

Output Class
Target Class

	1	2	3	4	5	6	7	8	9	
1	1426 13.1%	13 0.1%	0 0.0%	7 0.1%	8 0.1%	29 0.3%	2 0.0%	49 0.5%	8 0.1%	92.5% 7.5%
2	12 0.1%	2388 22.0%	1 0.0%	0 0.0%	1 0.0%	9 0.1%	1 0.0%	11 0.1%	26 0.2%	97.5% 2.5%
3	3 0.0%	1 0.0%	2935 27.0%	2 0.0%	0 0.0%	1 0.0%	3 0.0%	4 0.0%	0 0.0%	99.5% 0.5%
4	8 0.1%	2 0.0%	0 0.0%	455 4.2%	6 0.1%	5 0.0%	0 0.0%	19 0.2%	1 0.0%	91.7% 8.3%
5	2 0.0%	2 0.0%	0 0.0%	2 0.0%	18 0.2%	0 0.0%	0 0.0%	8 0.1%	0 0.0%	56.3% 43.8%
6	25 0.2%	7 0.1%	0 0.0%	2 0.0%	4 0.0%	679 6.2%	0 0.0%	21 0.2%	19 0.2%	89.7% 10.3%
7	2 0.0%	2 0.0%	2 0.0%	1 0.0%	2 0.0%	2 0.0%	387 3.6%	1 0.0%	1 0.0%	96.8% 3.2%
8	43 0.4%	19 0.2%	4 0.0%	3 0.0%	3 0.0%	21 0.2%	4 0.0%	1103 10.1%	3 0.0%	91.7% 8.3%
9	21 0.2%	44 0.4%	0 0.0%	3 0.0%	0 0.0%	5 0.0%	1 0.0%	12 0.1%	955 8.8%	91.7% 8.3%
	92.5% 7.5%	96.4% 3.6%	99.8% 0.2%	95.8% 4.2%	42.9% 57.1%	90.4% 9.6%	97.2% 2.8%	89.8% 10.2%	94.3% 5.7%	95.2% 4.8%

The polynomial Kernel is a variation of the quadratic kernel except that the order of the polynomial is not fixed.

However this has a comparatively higher accuracy compared to the other kernels, especially in the case of class V

4. **Neural Networks:**

We are using an 80/20 approach i.e splitting the entire data set into 80% of actual data and using it for training and then using the rest 20% for testing.

While using Neural networks there are multiple features that can be altered to acquire accuracy, such as:

1. Number of Neurons on each layer
2. Learning Rate
3. Number of iterations
4. Number of layers

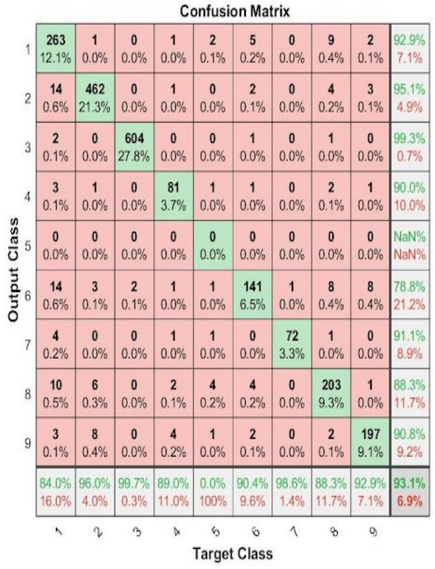
The attached figures in the next document have the following number of neurons on each layer

Learning rate = 0.2

Number of itrs = 10000

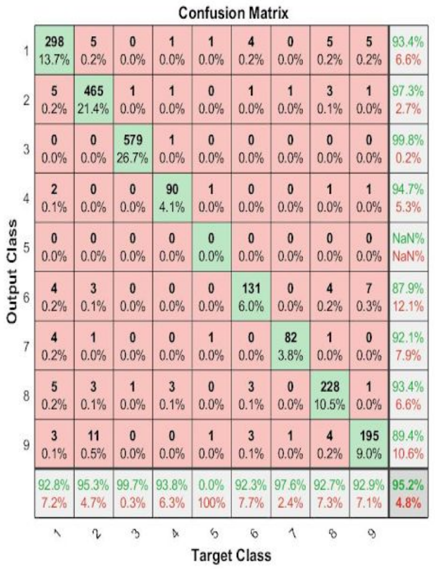
Fig 1	S2 = 50	S3 = 10	Accuracy = 93.1
Fig2	S2 = 100	S3 = 50	94.6
fig3	S2 = 350	S2 = 250	92.7
Fig 4	S2 = 400	S3 = 300	96.3
fig5	S2 = 500	S3 = 400	95.7

Number of Neurons on each layer


Confusion Matrix

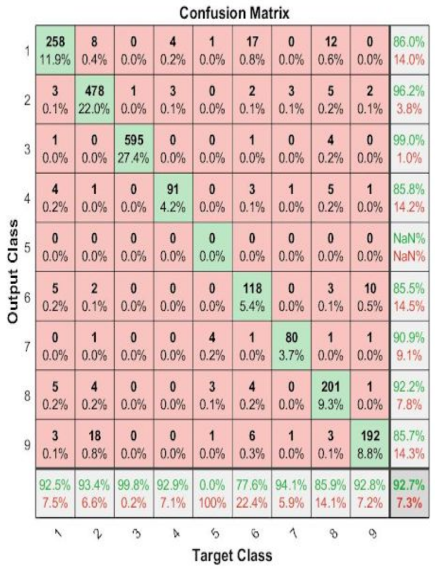
	1	2	3	4	5	6	7	8	9	
1	263 12.1%	1 0.0%	0 0.0%	1 0.0%	2 0.1%	5 0.2%	0 0.0%	9 0.4%	2 0.1%	92.9% 7.1%
2	14 0.6%	462 21.3%	0 0.0%	1 0.0%	0 0.0%	2 0.1%	0 0.0%	4 0.2%	3 0.1%	95.1% 4.9%
3	2 0.1%	0 0.0%	604 27.8%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	1 0.0%	0 0.0%	99.3% 0.7%
4	3 0.1%	1 0.0%	0 0.0%	81 3.7%	1 0.0%	1 0.0%	0 0.0%	2 0.1%	1 0.0%	90.0% 10.0%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
6	14 0.6%	3 0.1%	2 0.1%	1 0.0%	1 0.0%	141 6.5%	1 0.0%	8 0.4%	8 0.4%	78.8% 21.2%
7	4 0.2%	0 0.0%	0 0.0%	1 0.0%	1 0.0%	0 0.0%	72 3.3%	1 0.0%	0 0.0%	91.1% 8.9%
8	10 0.5%	6 0.3%	0 0.0%	2 0.1%	4 0.2%	4 0.2%	0 0.0%	203 9.3%	1 0.0%	88.3% 11.7%
9	3 0.1%	8 0.4%	0 0.0%	4 0.2%	1 0.0%	2 0.1%	0 0.0%	2 0.1%	197 9.1%	90.8% 9.2%
	84.0% 16.0%	96.0% 4.0%	99.7% 0.3%	89.0% 11.0%	0.0% 100%	90.4% 9.6%	98.6% 1.4%	88.3% 11.7%	92.9% 7.1%	93.1% 6.9%

Output Class
Target Class


Confusion Matrix

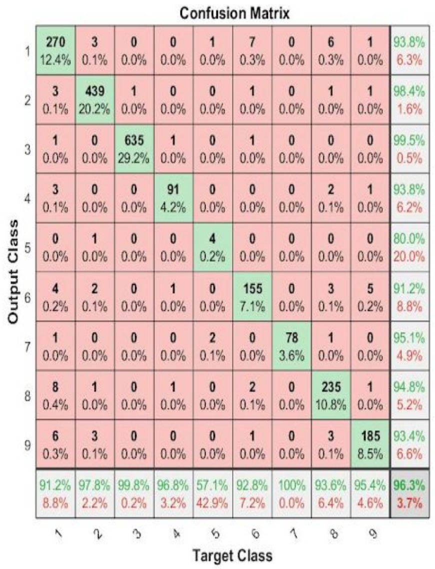
	1	2	3	4	5	6	7	8	9	
1	298 13.7%	5 0.2%	0 0.0%	1 0.0%	1 0.0%	4 0.2%	0 0.0%	5 0.2%	5 0.2%	93.4% 6.6%
2	5 0.2%	465 21.4%	1 0.0%	1 0.0%	0 0.0%	1 0.0%	1 0.0%	3 0.1%	1 0.0%	97.3% 2.7%
3	0 0.0%	0 0.0%	579 26.7%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	99.8% 0.2%
4	2 0.1%	0 0.0%	0 0.0%	90 4.1%	1 0.0%	0 0.0%	0 0.0%	1 0.0%	1 0.0%	94.7% 5.3%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
6	4 0.2%	3 0.1%	0 0.0%	0 0.0%	0 0.0%	131 6.0%	0 0.0%	4 0.2%	7 0.3%	87.9% 12.1%
7	4 0.2%	1 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	82 3.8%	1 0.0%	0 0.0%	92.1% 7.9%
8	5 0.2%	3 0.1%	1 0.0%	3 0.1%	0 0.0%	3 0.1%	0 0.0%	228 10.5%	1 0.0%	93.4% 6.6%
9	3 0.1%	11 0.5%	0 0.0%	0 0.0%	1 0.0%	3 0.1%	1 0.0%	4 0.2%	195 9.0%	89.4% 10.6%
	92.8% 7.2%	95.3% 4.7%	99.7% 0.3%	93.8% 6.3%	0.0% 100%	92.3% 7.7%	97.6% 2.4%	92.7% 7.3%	92.9% 7.1%	95.2% 4.8%

Output Class
Target Class


Confusion Matrix

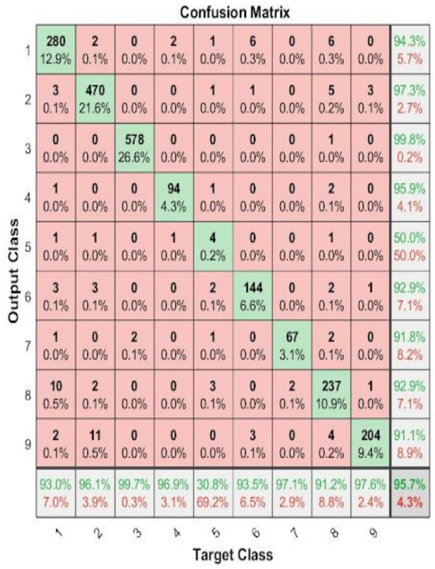
	1	2	3	4	5	6	7	8	9	
1	258 11.9%	8 0.4%	0 0.0%	4 0.2%	1 0.0%	17 0.8%	0 0.0%	12 0.6%	0 0.0%	86.0% 14.0%
2	3 0.1%	478 22.0%	1 0.0%	3 0.1%	0 0.0%	2 0.1%	3 0.1%	5 0.2%	2 0.1%	96.2% 3.8%
3	1 0.0%	0 0.0%	595 27.4%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	4 0.2%	0 0.0%	99.0% 1.0%
4	4 0.2%	1 0.0%	0 0.0%	91 4.2%	0 0.0%	3 0.1%	1 0.0%	5 0.2%	1 0.0%	85.8% 14.2%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
6	5 0.2%	2 0.1%	0 0.0%	0 0.0%	0 0.0%	118 5.4%	0 0.0%	3 0.1%	10 0.5%	85.5% 14.5%
7	0 0.0%	1 0.0%	0 0.0%	0 0.0%	4 0.2%	1 0.0%	80 3.7%	1 0.0%	1 0.0%	90.9% 9.1%
8	5 0.2%	4 0.2%	0 0.0%	0 0.0%	3 0.1%	4 0.2%	0 0.0%	201 9.3%	1 0.0%	92.2% 7.8%
9	3 0.1%	18 0.8%	0 0.0%	0 0.0%	1 0.0%	6 0.3%	1 0.0%	3 0.1%	192 8.8%	85.7% 14.3%
	92.5% 7.5%	93.4% 6.6%	99.8% 0.2%	92.9% 7.1%	0.0% 100%	77.6% 22.4%	94.1% 5.9%	85.9% 14.1%	92.8% 7.2%	92.7% 7.3%

Output Class
Target Class


Confusion Matrix

	1	2	3	4	5	6	7	8	9	
1	270 12.4%	3 0.1%	0 0.0%	0 0.0%	1 0.0%	7 0.3%	0 0.0%	6 0.3%	1 0.0%	93.8% 6.3%
2	3 0.1%	439 20.2%	1 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	1 0.0%	1 0.0%	98.4% 1.6%
3	1 0.0%	0 0.0%	635 29.2%	1 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	99.5% 0.5%
4	3 0.1%	0 0.0%	0 0.0%	91 4.2%	0 0.0%	0 0.0%	0 0.0%	2 0.1%	1 0.0%	93.8% 6.2%
5	0 0.0%	1 0.0%	0 0.0%	0 0.0%	4 0.2%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	80.0% 20.0%
6	4 0.2%	2 0.1%	0 0.0%	1 0.0%	0 0.0%	155 7.1%	0 0.0%	3 0.1%	5 0.2%	91.2% 8.8%
7	1 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.1%	0 0.0%	78 3.6%	1 0.0%	0 0.0%	95.1% 4.9%
8	8 0.4%	1 0.0%	0 0.0%	1 0.0%	0 0.0%	2 0.1%	0 0.0%	235 10.8%	1 0.0%	94.8% 5.2%
9	6 0.3%	3 0.1%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	3 0.1%	185 8.5%	93.4% 6.6%
	91.2% 8.8%	97.8% 2.2%	99.8% 0.2%	96.8% 3.2%	57.1% 42.9%	92.8% 7.2%	100% 0.0%	93.6% 6.4%	95.4% 4.6%	96.3% 3.7%

Output Class
Target Class


Confusion Matrix

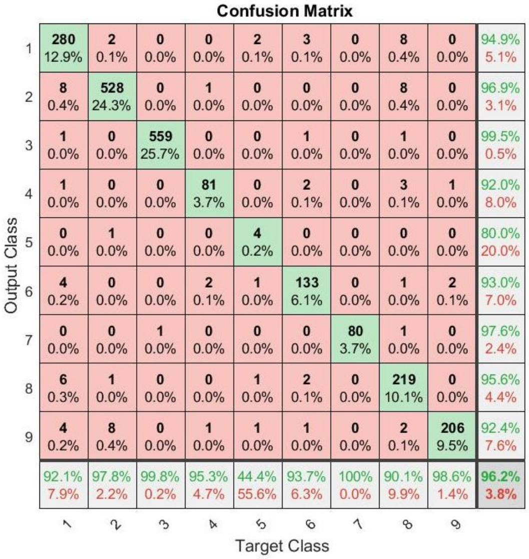
	1	2	3	4	5	6	7	8	9	
1	280 12.9%	2 0.1%	0 0.0%	2 0.1%	1 0.0%	6 0.3%	0 0.0%	6 0.3%	0 0.0%	94.3% 5.7%
2	3 0.1%	470 21.6%	0 0.0%	0 0.0%	1 0.0%	1 0.0%	0 0.0%	5 0.2%	3 0.1%	97.3% 2.7%
3	0 0.0%	0 0.0%	578 26.6%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	99.8% 0.2%
4	1 0.0%	0 0.0%	0 0.0%	94 4.3%	1 0.0%	0 0.0%	0 0.0%	2 0.1%	0 0.0%	95.9% 4.1%
5	1 0.0%	1 0.0%	0 0.0%	1 0.0%	4 0.2%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	50.0% 50.0%
6	3 0.1%	3 0.1%	0 0.0%	0 0.0%	2 0.1%	144 6.6%	0 0.0%	2 0.1%	1 0.0%	92.9% 7.1%
7	1 0.0%	0 0.0%	2 0.1%	0 0.0%	1 0.0%	0 0.0%	67 3.1%	2 0.1%	0 0.0%	91.8% 8.2%
8	10 0.5%	2 0.1%	0 0.0%	0 0.0%	3 0.1%	0 0.0%	2 0.1%	237 10.9%	1 0.0%	92.9% 7.1%
9	2 0.1%	11 0.5%	0 0.0%	0 0.0%	0 0.0%	3 0.1%	0 0.0%	4 0.2%	204 9.4%	91.1% 8.9%
	93.0% 7.0%	96.1% 3.9%	99.7% 0.3%	96.9% 3.1%	30.8% 69.2%	93.5% 6.5%	97.1% 2.9%	91.2% 8.8%	97.6% 2.4%	95.7% 4.3%

It is clear that the number of correct neurons for maximum result is S2 = 400 and s3 = 300.

Even though the other numbers are close we pick the above number since it has higher classification of class V.

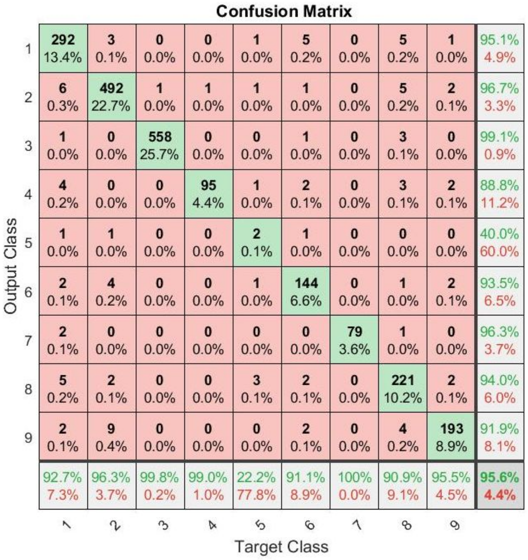
Now we can vary other features such as number of iterations and learning rate.

Learning rate:



Confusion Matrix

Output Class \ Target Class	1	2	3	4	5	6	7	8	9	
1	280 12.9%	2 0.1%	0 0.0%	0 0.0%	2 0.1%	3 0.1%	0 0.0%	8 0.4%	0 0.0%	94.9% 5.1%
2	8 0.4%	528 24.3%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	8 0.4%	0 0.0%	96.9% 3.1%
3	1 0.0%	0 0.0%	559 25.7%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	1 0.0%	0 0.0%	99.5% 0.5%
4	1 0.0%	0 0.0%	0 0.0%	81 3.7%	0 0.0%	2 0.1%	0 0.0%	3 0.1%	1 0.0%	92.0% 8.0%
5	0 0.0%	1 0.0%	0 0.0%	0 0.0%	4 0.2%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	80.0% 20.0%
6	4 0.2%	0 0.0%	0 0.0%	2 0.1%	1 0.0%	133 6.1%	0 0.0%	1 0.0%	2 0.1%	93.0% 7.0%
7	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	80 3.7%	1 0.0%	0 0.0%	97.6% 2.4%
8	6 0.3%	1 0.0%	0 0.0%	0 0.0%	1 0.0%	2 0.1%	0 0.0%	219 10.1%	0 0.0%	95.6% 4.4%
9	4 0.2%	8 0.4%	0 0.0%	1 0.0%	1 0.0%	1 0.0%	0 0.0%	2 0.1%	206 9.5%	92.4% 7.6%
	92.1% 7.9%	97.8% 2.2%	99.8% 0.2%	95.3% 4.7%	44.4% 55.6%	93.7% 6.3%	100% 0.0%	90.1% 9.9%	98.6% 1.4%	96.2% 3.8%



Confusion Matrix

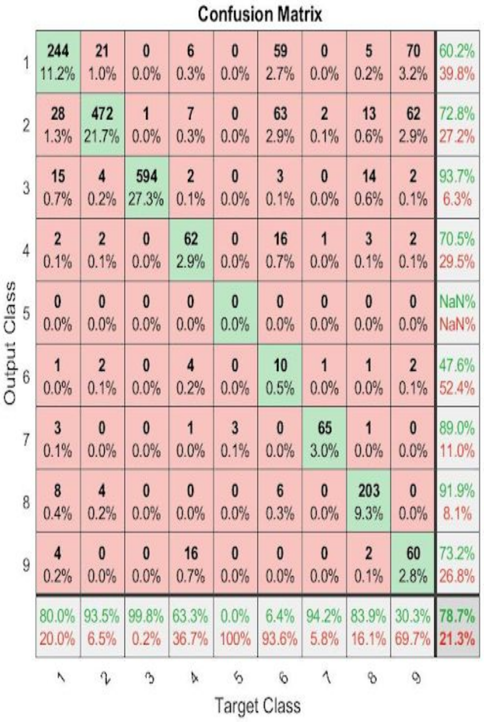
Output Class \ Target Class	1	2	3	4	5	6	7	8	9	
1	292 13.4%	3 0.1%	0 0.0%	0 0.0%	1 0.0%	5 0.2%	0 0.0%	5 0.2%	1 0.0%	95.1% 4.9%
2	6 0.3%	492 22.7%	1 0.0%	1 0.0%	1 0.0%	1 0.0%	0 0.0%	5 0.2%	2 0.1%	96.7% 3.3%
3	1 0.0%	0 0.0%	558 25.7%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	3 0.1%	0 0.0%	99.1% 0.9%
4	4 0.2%	0 0.0%	0 0.0%	95 4.4%	1 0.0%	2 0.1%	0 0.0%	3 0.1%	2 0.1%	88.8% 11.2%
5	1 0.0%	1 0.0%	0 0.0%	0 0.0%	2 0.1%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	40.0% 60.0%
6	2 0.1%	4 0.2%	0 0.0%	0 0.0%	1 0.0%	144 6.6%	0 0.0%	1 0.0%	2 0.1%	93.5% 6.5%
7	2 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	79 3.6%	1 0.0%	0 0.0%	96.3% 3.7%
8	5 0.2%	2 0.1%	0 0.0%	0 0.0%	3 0.1%	2 0.1%	0 0.0%	221 10.2%	2 0.1%	94.0% 6.0%
9	2 0.1%	9 0.4%	0 0.0%	0 0.0%	0 0.0%	2 0.1%	0 0.0%	4 0.2%	193 8.9%	91.9% 8.1%
	92.7% 7.3%	96.3% 3.7%	99.8% 0.2%	99.0% 1.0%	22.2% 77.8%	91.1% 8.9%	100% 0.0%	90.9% 9.1%	95.5% 4.5%	95.6% 4.4%

The learning rate of the image on left is 0.2 and right is 1,

We can clearly see a drop in the accuracy aswell as classification of class V.

This can be attributed to the steep learning rate which can miss a few data points on the curve.

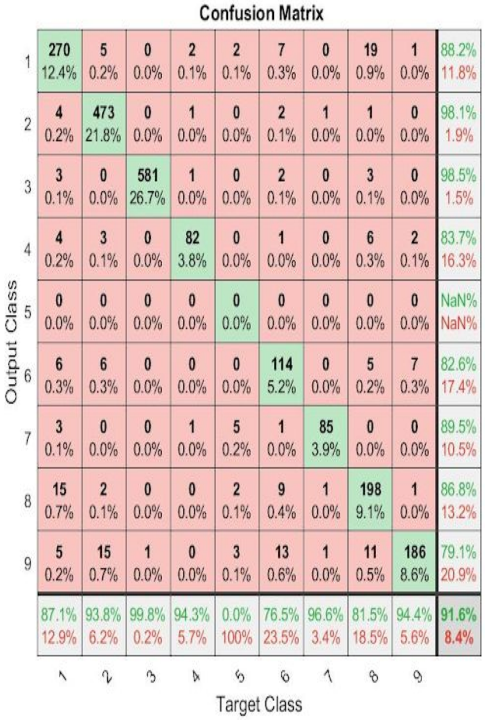
Number of Iterations



Confusion Matrix

Output Class / Target Class

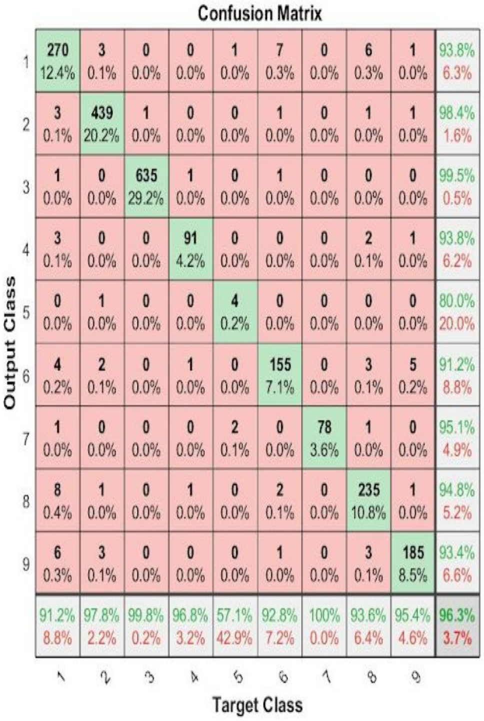
	1	2	3	4	5	6	7	8	9	
1	244 11.2%	21 1.0%	0 0.0%	6 0.3%	0 0.0%	59 2.7%	0 0.0%	5 0.2%	70 3.2%	60.2% 39.8%
2	28 1.3%	472 21.7%	1 0.0%	7 0.3%	0 0.0%	63 2.9%	2 0.1%	13 0.6%	62 2.9%	72.8% 27.2%
3	15 0.7%	4 0.2%	594 27.3%	2 0.1%	0 0.0%	3 0.1%	0 0.0%	14 0.6%	2 0.1%	93.7% 6.3%
4	2 0.1%	2 0.1%	0 0.0%	62 2.9%	0 0.0%	16 0.7%	1 0.0%	3 0.1%	2 0.1%	70.5% 29.5%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
6	1 0.0%	2 0.1%	0 0.0%	4 0.2%	0 0.0%	10 0.5%	1 0.0%	1 0.0%	2 0.1%	47.6% 52.4%
7	3 0.1%	0 0.0%	0 0.0%	1 0.0%	3 0.1%	0 0.0%	65 3.0%	1 0.0%	0 0.0%	89.0% 11.0%
8	8 0.4%	4 0.2%	0 0.0%	0 0.0%	0 0.0%	6 0.3%	0 0.0%	203 9.3%	0 0.0%	91.9% 8.1%
9	4 0.2%	0 0.0%	0 0.0%	16 0.7%	0 0.0%	0 0.0%	0 0.0%	2 0.1%	60 2.8%	73.2% 26.8%
	80.0% 20.0%	93.5% 6.5%	99.8% 0.2%	63.3% 36.7%	0.0% 100%	6.4% 93.6%	94.2% 5.8%	83.9% 16.1%	30.3% 69.7%	78.7% 21.3%



Confusion Matrix

Output Class / Target Class

	1	2	3	4	5	6	7	8	9	
1	270 12.4%	5 0.2%	0 0.0%	2 0.1%	2 0.1%	7 0.3%	0 0.0%	19 0.9%	1 0.0%	88.2% 11.8%
2	4 0.2%	473 21.8%	0 0.0%	1 0.0%	0 0.0%	2 0.1%	1 0.0%	1 0.0%	0 0.0%	98.1% 1.9%
3	3 0.1%	0 0.0%	581 26.7%	1 0.0%	0 0.0%	2 0.1%	0 0.0%	3 0.1%	0 0.0%	98.5% 1.5%
4	4 0.2%	3 0.1%	0 0.0%	82 3.8%	0 0.0%	1 0.0%	0 0.0%	6 0.3%	2 0.1%	83.7% 16.3%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
6	6 0.3%	6 0.3%	0 0.0%	0 0.0%	0 0.0%	114 5.2%	0 0.0%	5 0.2%	7 0.3%	82.6% 17.4%
7	3 0.1%	0 0.0%	0 0.0%	1 0.0%	5 0.2%	1 0.0%	85 3.9%	0 0.0%	0 0.0%	89.5% 10.5%
8	15 0.7%	2 0.1%	0 0.0%	0 0.0%	2 0.1%	9 0.4%	1 0.0%	198 9.1%	1 0.0%	86.8% 13.2%
9	5 0.2%	15 0.7%	1 0.0%	0 0.0%	3 0.1%	13 0.6%	1 0.0%	11 0.5%	186 8.6%	79.1% 20.9%
	87.1% 12.9%	93.8% 6.2%	99.8% 0.2%	94.3% 5.7%	0.0% 100%	76.5% 23.5%	96.6% 3.4%	81.5% 18.5%	94.4% 5.6%	91.6% 8.4%



Confusion Matrix

Output Class / Target Class

	1	2	3	4	5	6	7	8	9	
1	270 12.4%	3 0.1%	0 0.0%	0 0.0%	1 0.0%	7 0.3%	0 0.0%	6 0.3%	1 0.0%	93.8% 6.3%
2	3 0.1%	439 20.2%	1 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	1 0.0%	1 0.0%	98.4% 1.6%
3	1 0.0%	0 0.0%	635 29.2%	1 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	99.5% 0.5%
4	3 0.1%	0 0.0%	0 0.0%	91 4.2%	0 0.0%	0 0.0%	0 0.0%	2 0.1%	1 0.0%	93.8% 6.2%
5	0 0.0%	1 0.0%	0 0.0%	0 0.0%	4 0.2%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	80.0% 20.0%
6	4 0.2%	2 0.1%	0 0.0%	1 0.0%	0 0.0%	155 7.1%	0 0.0%	3 0.1%	5 0.2%	91.2% 8.8%
7	1 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.1%	0 0.0%	78 3.6%	1 0.0%	0 0.0%	95.1% 4.9%
8	8 0.4%	1 0.0%	0 0.0%	1 0.0%	0 0.0%	2 0.1%	0 0.0%	235 10.8%	1 0.0%	94.8% 5.2%
9	6 0.3%	3 0.1%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	3 0.1%	185 8.5%	93.4% 6.6%
	91.2% 8.8%	97.8% 2.2%	99.8% 0.2%	96.8% 3.2%	57.1% 42.9%	92.8% 7.2%	100% 0.0%	93.6% 6.4%	95.4% 4.6%	96.3% 3.7%

The above figures have 100, 1000 and 10,000 iterations respectively

We can clearly see the accuracy increasing with higher training.

The lower accuracy in lower number of iterations can be attributed to the fact that the model has not been trained properly to classify all possible classes

Solution/Suggestions to improve the accuracy

- The main issue pertaining to this data set is the difficulty in classifying the class V(SIMDA) type of files, the easiest way to improve accuracy would be to get more data of this type.

- With the given data set however multiple methods can be adopted,one such method is:
  - A multifold method where a number of data points equal to the lowest available class of data(in this case SIMDA) can be taken.
  - This can be clubbed with SIMDA and the model can be trained.
  - In the similar fashion the entire dataset could be trained negating the previously taken values to get a better output

- The false positive and false negative cases can be gathered and the model can be trained again with these cases and then retested to achieve better results