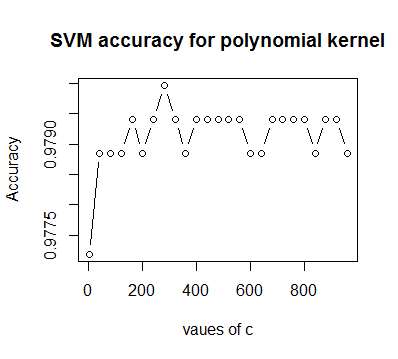
**Information – SL Assignment (Neural Networks)**

**OptDigits Dataset:**

* Status - Completed.
* It was an easy implementation of SMO on the OptDigits dataset. I have first given indices to all the attributes and labelled the class attribute as Target so that WEKA can understand it.
* SMO - implements John Platt's sequential minimal optimization algorithm for training a support vector classifier.
* I have trained and have predicted the class values using SVM for different kernels and respective logistic calibrator.
* Different kernels included
  + Normalized Poly Kernel
  + Poly Kernel
  + PUK Kernel
  + RBF Kernel
* I have calculated classifiers outputs for different C values and cross-validated to get the best fit for the training data and used the best fit model to predict values from the provided test data.
* The following is a plotted graph varying C values for the Poly Kernel Model SVM.
* 
* I have also checked if normalizing or standardizing the data is affecting the prediction accuracy and following summaries for standardized data and “No normalization or standardization” data show us the difference.
* Standardized SVM with RBF kernel for C=5

=== Summary ===

Correctly Classified Instances 1749 97.3289 %

Incorrectly Classified Instances 48 2.6711 %

Kappa statistic 0.9703

Mean absolute error 0.1603

Root mean squared error 0.2723

Relative absolute error 89.0352 %

Root relative squared error 90.782 %

Total Number of Instances 1797

* No Standardizing or Normalizing of data, SVM with RBF Kernel C=5.

=== Summary ===

Correctly Classified Instances 1354 75.3478 %

Incorrectly Classified Instances 443 24.6522 %

Kappa statistic 0.7263

Mean absolute error 0.1635

Root mean squared error 0.2785

Relative absolute error 90.8146 %

Root relative squared error 92.8193 %

Total Number of Instances 1797

* Overall I have found that while almost all the kernels were giving predictions over 96%, using cross-validation made me choose the right parameters for the training data and avoided overfitting of the data. Which made it possible to get very high accuracies when predicting with the provided Test Data.

**Amazon Reviews Dataset:**

* Status – Completed.
* The challenging part dealing with Amazon Reviews Dataset is to understand how to classify text data.
* Implemented Bag of Words algorithm to the reviews column of the dataset. Cleaned the data from stopwords (using NlTK’s corpus – stopwords), numbers and converted to a set of meaningful words.
* Count Vectorizer in scikit learn was used to create a bag of words model which will return binary value if that is word present in each review of the data set or not. The same will be iterative for all the words in the bag of words.
* Later I learned of something called TF-IDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval, text mining, and user modelling.
* Instead of using count vectorizer I used TF-IDF vectorizer in scikit-learn to give a vector for each review with which I performed decision tree classification.
* I used the SVC (Support Vector Classifier) classifier from the scikit-learn’s Support Vector Machine library.
* The default parameters for the default sklearn’s SVC classifier were

SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape=None, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, probability=False, random\_state=None, shrinking=True,

tol=0.001, verbose=False)

* I have done analysis using multiple kernels, C-values, different Gamma values, varying the weighting function and using different polynomial degrees for poly kernel.
* I have found that poly kernel was giving me predictions for only one class. Which means that it was particularly biased on only one class value. It proved that poly kernel was not that great when it is used for the vector data for text classification. Different degrees of polynomial returned the same issue.
* We can see the bad results using the poly kernel

[2 2 2 ..., 2 2 2]

0.0603015075377

[[ 0 163 0 0 0]

[ 0 120 0 0 0]

[ 0 174 0 0 0]

[ 0 344 0 0 0]

[ 0 1189 0 0 0]]

* Accuracy and Confusion matrix for RBF Kernel for C = 1.0 and Gamma = 1.0:

0.636180904523

[[ 43 0 1 4 115]

[ 18 0 2 5 95]

[ 2 1 6 16 149]

[ 2 0 1 37 304]

[ 2 0 0 7 1180]]

* Accuracy and Confusion matrix for RBF Kernel for C = 1.0 and Gamma = 1.0:

0.653266331658

[[ 65 9 10 10 69]

[ 28 1 17 13 61]

[ 6 5 23 33 107]

[ 3 0 14 88 239]

[ 6 2 7 51 1123]]

* We can see that increasing C value avoids misclassifying the data, similarly increasing it until the Margin is small enough to avoid misclassification and large enough to provide classification is important.
* I have therefore tried multiple class values and checked it using cross-validation, helped me determine the best C value to fit for the training data, I used the same model for prediction for the provided test data.
* I have done analysis using multiple kernels:
  + RBF Kernel
  + Linear Kernel
  + Poly Kernel
  + Sigmoid Kernel
* Typically, the best results are given when we vary the gamma, C values and use cross-validation as a measure to get the best fit and later use the same model for prediction using the test data.
* We can also see that finding the best combination is important, not just varying one parameter.
* RBF kernel C=1.0 and Gamma=0.1

0.576884422111

[[ 76 25 22 19 21]

[ 38 14 28 17 23]

[ 15 25 38 42 54]

[ 12 12 42 106 172]

[ 31 30 63 151 914]]

* RBF Kernel C=1.0 and Gamma=0.1

0.636180904523

[[ 43 0 1 4 115]

[ 18 0 2 5 95]

[ 2 1 6 16 149]

[ 2 0 1 37 304]

[ 2 0 0 7 1180]]

* We can see that just varying the gamma value from 1.0 to 0.1 has given us accuracy boost of about 6%.
* I was able to get a maximum accuracy of 65% for the following two parameter combinations after trying multiple kernels, C values and Gamma values.
* Sigmoid C10 Gamma=1.0

0.65175879397

[[ 64 6 13 9 71]

[ 34 2 12 11 61]

[ 9 5 22 33 105]

[ 5 1 12 78 248]

[ 7 1 10 40 1131]]

* RBF C=10 Gamma=1.0

0.653266331658

[[ 65 9 10 10 69]

[ 28 1 17 13 61]

[ 6 5 23 33 107]

[ 3 0 14 88 239]

[ 6 2 7 51 1123]]