Homework 5 : Statistical Data Mining (EAS 506)

Shubham Sharma, Person No.: 50290293, Class No.: 44

December 17, 2018

Question 1: Fit a series of random-forest classifiers to the SPAM data, to explore the sensitivity to m (the number of randomly selected inputs for each tree). Plot both the OOB error as well as the test error against a suitably chosen range of values for m.

The training and test set data is split in the ratio 2:1. The plot for the OOB error as well as the test error against a suitably chosen range of values for m is shown in figure 1.

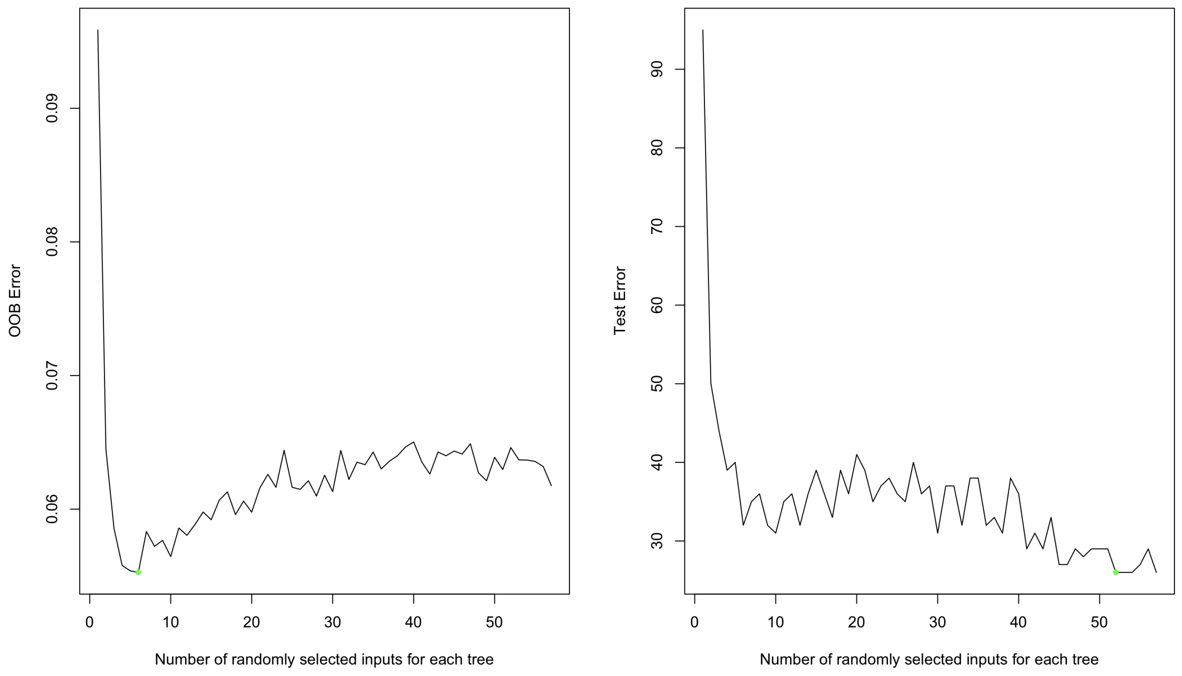


Figure 1: Plot for the OOB error as well as the test error against a suitably chosen range of values for m

The minimum OOB error of 0.0553 is obtained at m = 6 while a minimum test error of 26 is obtained at m = 52.

Question 2: Fit a neural network to the spam data of Section 9.1.2. The data is available through the package ElemStatLearn. Use cross-validation or the hold out method to determine the number of neurons to use in the layer. Compare your results to those for the additive model given in the chapter. When making the comparison, consider both the classification performance and interpretability of the final model.

The response variable 'spam' has been converted to a binary variable with values 1 for spam and 0 for email. The training and test set data is split in the ratio 2:1.

The threshold value for the partial derivatives of the error function is taken as 0.15 to optimize the algorithm. Since, this is a classification problem, the error function is taken to be cross-entropy(deviance). The argument linear.output is taken to be FALSE which causes the activation function to be applied which is used to make the result of the cross product of the covariate or neurons and the weights smooth. The plot of the generalization(test set) error with the number of neurons in the hidden layer is shown in figure 2. The minimum test set error is obtained for 4 neurons in the hidden layer and the value for the same is 0.06193. Thus, for the given model 4 neurons should be used in the hidden layer although we get a reasonable low error for 2 neurons as well. Thus, to save computation 2 neurons is also a good choice.

The graphical representation of the model is shown in figure 3 with weights on each connection.

The generalized additive model which is fit in chapter 9 has a test error rate of 5.5 % for the additive logistic regression model whereas for the linear logistic regression, it has a test error rate of 7.6 %. Thus, we see that in this case the additive model performed slightly better than the neural network model although this might also depend on the kind of train-test split taken for the additive model. The major advantage here is the interpretability of the additive model. Neural networks are like black boxes i.e. their outcome is much more difficult than explaining the outcome of simpler model such as the additive model. On the other hand, for the additive model the contribution of each variable can be seen and decomposed into a linear component and the remaining nonlinear component. The correlations of each predictor with the response variable is also obtained although the additive model might not feasible when a large number of predictors are there.

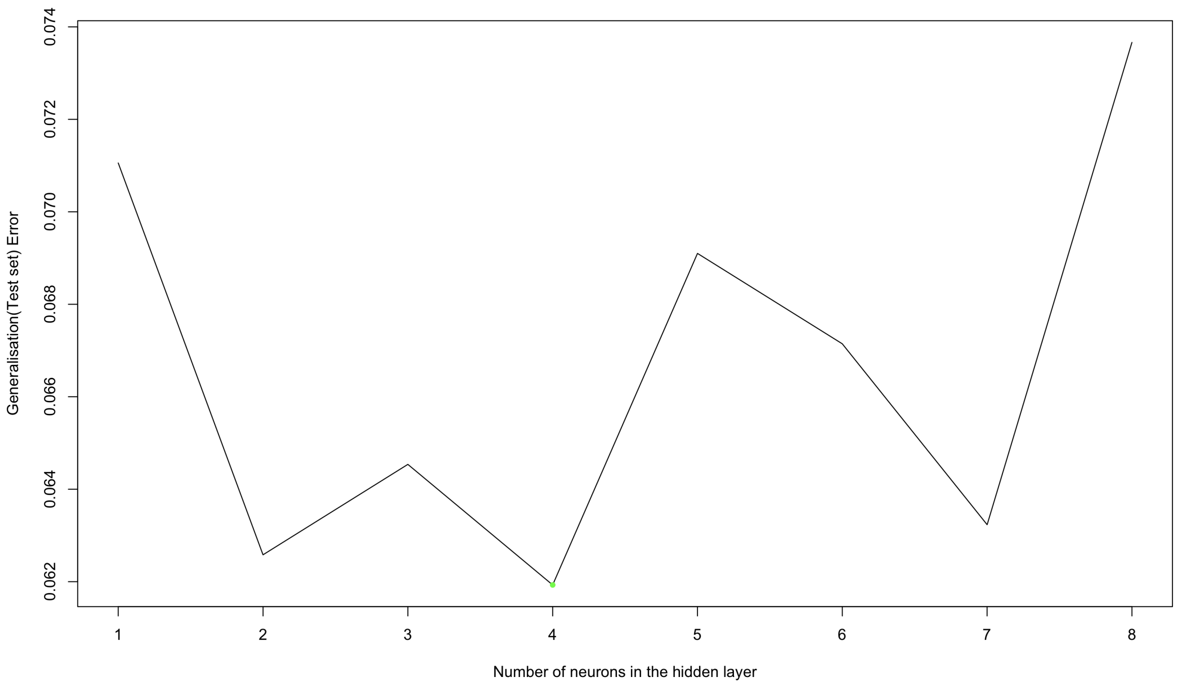


Figure 2: Generalisation (Test Set) error vs Number of neurons in the hidden layer

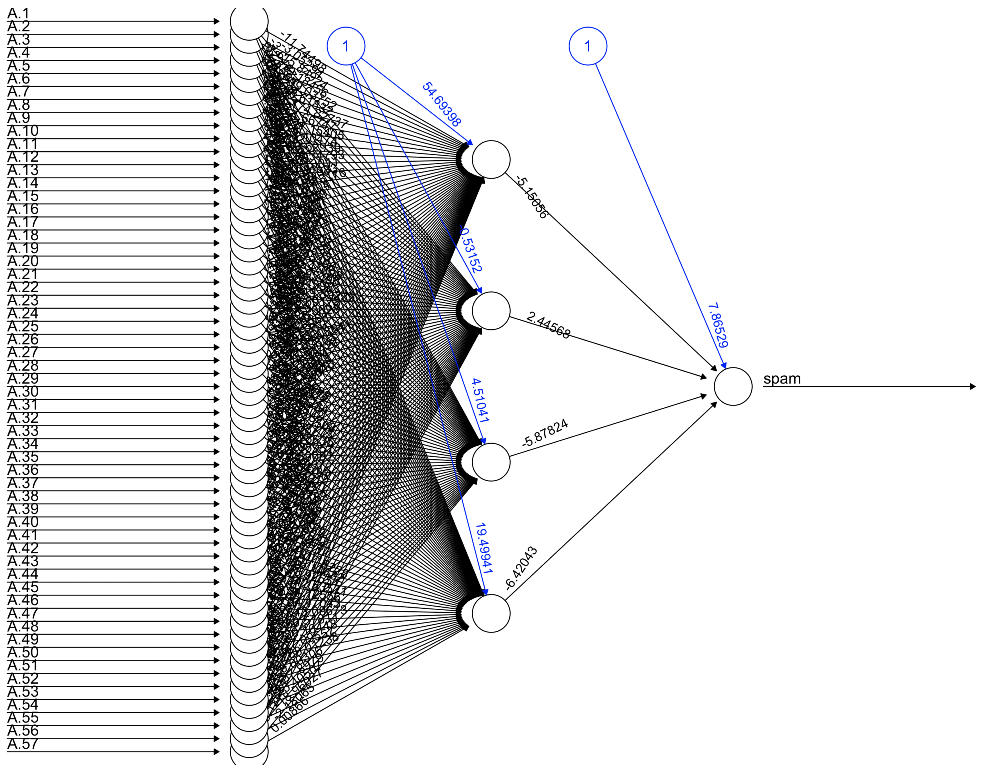


Figure 3: Graphical Representation of the Neural Network model

Question 3: Take any classification data set and divide it up into a learning set and a test set. Change the value of one observation on one input variable in the learning set so that the value is now a univariate outlier. Fit separate single- hidden-layer neural networks to the original learning-set data and to the learning- set data with the outlier. Use cross-validation or the hold out method to determine the number of neurons to use in the layer. Comment on the effect of the outlier on the fit and on its effect on classifying the test set. Shrink the value of that outlier toward its original value and evaluate when the effect of the outlier on the fit vanishes. How far away must the outlier move from its original value that significant changes to the network coefficient estimates occur?

The dataset used is SPAM dataset from ElemStatLearn package. The training and test set data is split in the ratio 11:9. The best number of hidden neurons was obtained from the previous question, i.e. 4. Here we created, 2 sets of SPAM data one with the outlier effect i.e. the values of the SPAM2[5,1] was changed from the 0.25 to 10 and then the corresponding training and test sets were created and the neural net was applied and the error values were evaluated. Here,2 sets of SPAM data are created: one with the outlier effect i.e. the values of the SPAM2[5,1] was changed from the 0.25 to 10 and then the corresponding training and test sets were created and the neural net was applied and the error values were evaluated. The outlier changes the error by 0.38. The error of the outlier from 10 to 0.25 was reduced and it was observed where the effect of the outlier vanishes. From the above procedure we were able to get the errors as: 0.2873478856 and 0.6246196332 which shows the error increasing almost by 0.38 due to the addition of the outlier. The outlier value where its effect vanishes is 0, which happens to be the original value, i.e. until we dont reduce the difference of outlier value to its original variable to null, the effect are visible.

Question 4: This problem involves the OJ data set in the ISLR package. We are interested in the prediction of Purchase. Divide the data into test and training.

(A) Fit a support vector classifier with varying cost parameters over the range [0.01, 10]. Plot the training and test error across this spectrum of cost parameters, and determine the optimal cost.

(B) Repeat the exercise in (A) for a support vector machine with a radial kernel. (Use the default parameter for gamma). Repeat the exercise again for a support vector machine with a polynomial kernel of degree=2. Reflect on the performance of the SVM with different kernels, and the support vector classifier, i.e., SVM with a linear kernel.

The training and test set data is split in the ratio 7:3.

The test and training errors are summarized in table 1 for different kernels.

SVM Kernels	Train error	Test error	Cost for minimum test error
Linesr	0.1669	0.1776	0.01
Radial	0.1335	0.1776	4.01
Polynomial kernel with degree = 2	0.1388	0.1900	9.01

Table 1: Summary for different SVM Kernels

The radial kernel has better performance as compared to the quadratic kernel since its test error is lower and also the optimal cost is 4.01 whereas for quadratic, it is 9.01. The training errors for both non-linear classifiers is lower than the linear one. The optimal cost for test data is obtained at 0.01 for the linear kernel. The train and test error versus the cost is plotted for different kernels in figures 4, 5 and 6.

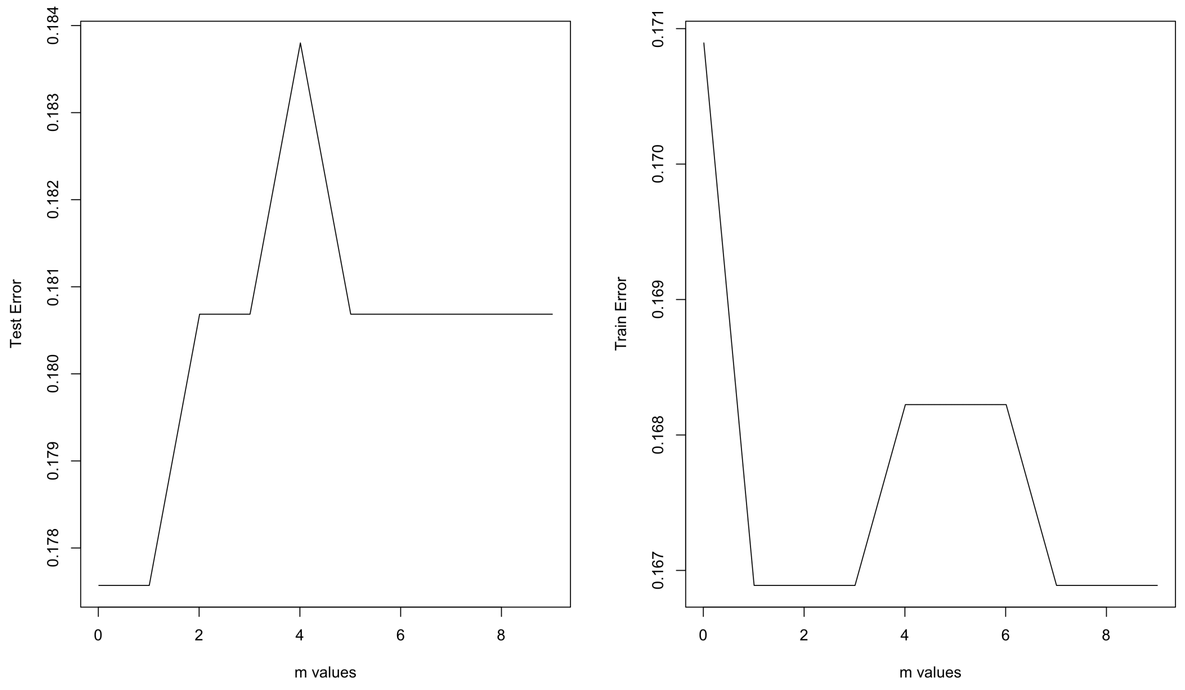


Figure 4: Test and Train error vs cost for Linear Kernel. Index of cost values represent costs: 0.01 1.01 2.01 3.01 4.01 5.01 6.01 7.01 8.01 9.01

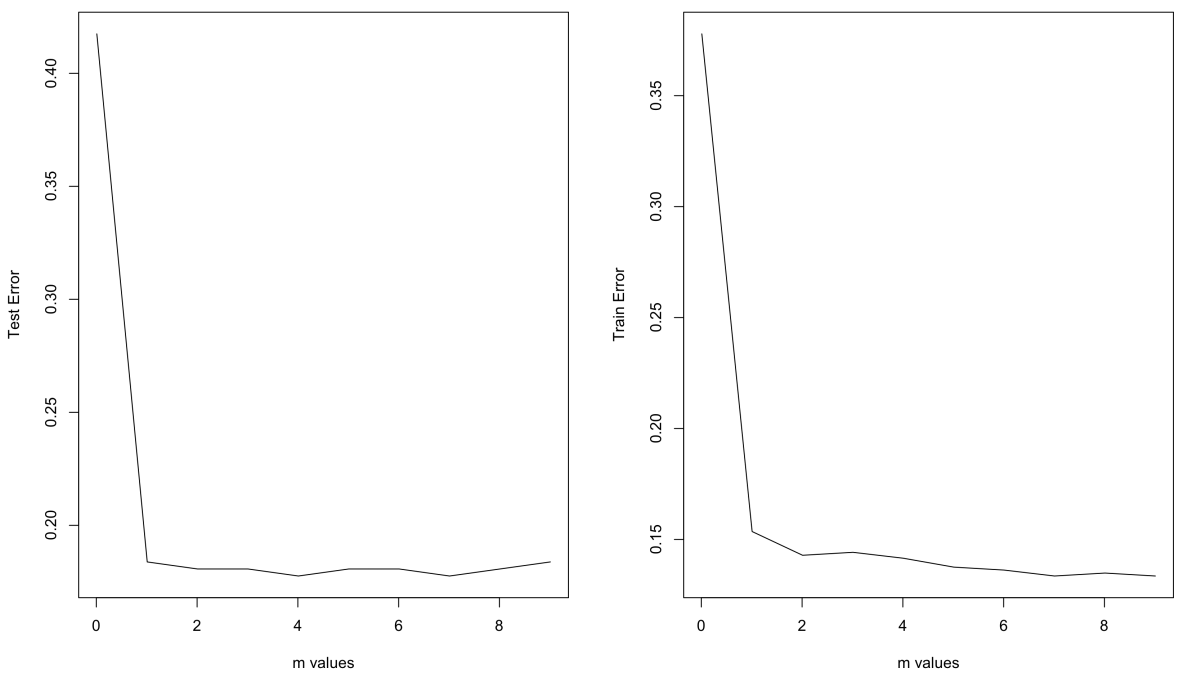


Figure 5: Test and Train error vs cost for Radial Kernel. Index of cost values represent costs: 0.01 1.01 2.01 3.01 4.01 5.01 6.01 7.01 8.01 9.01

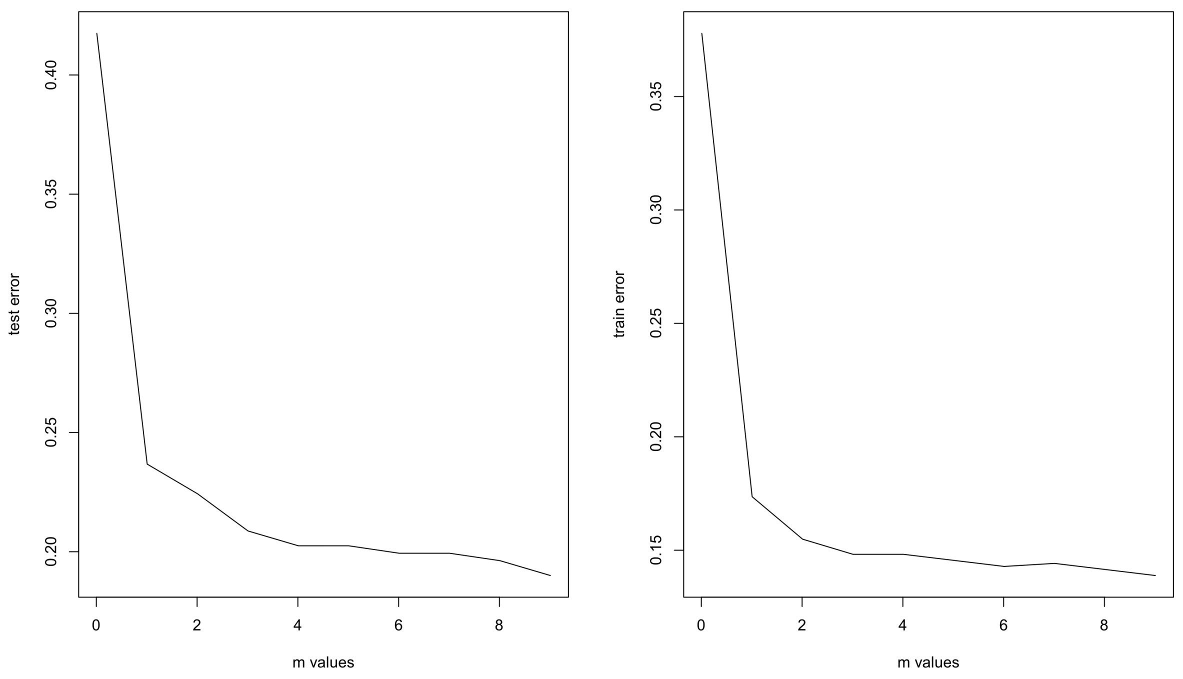
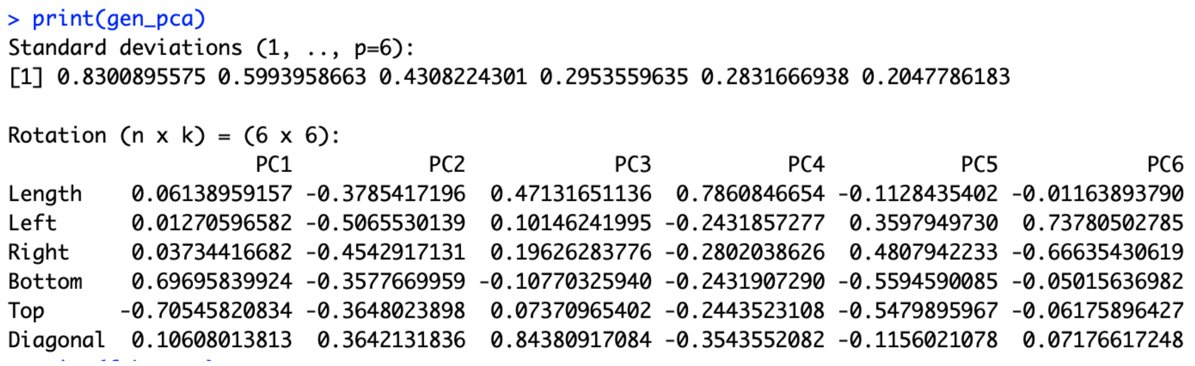


Figure 6: Test and Train error vs cost for Quadratic Kernel. Index of cost values represent costs: 0.01 1.01 2.01 3.01 4.01 5.01 6.01 7.01 8.01 9.01

Question 5: Access the SwissBankNotes data (posted with assignment). The data consists of six variables measured on 200 old Swiss 1,000-franc bank notes. The first 100 are genuine and the second 100 are counterfeit. The six variables are length of the bank note, height of the bank note, measured on the left, height of the bank note measured on the right, distance of the inner frame to the lower border, distance of inner frame to upper border, and length of the diagonal. Carry out a PCA of the 100 genuine bank notes, of the 100 counterfeit bank notes, and all of the 200 bank notes combined. Do you notice any differences in the results? Show all work in the selection of Principal Components, including diagnostic plots.

The data is taken from banknote dataset in mclust package. The attributes top and bottom margin are almost orthogonal to each other and thus we can say that they have a correlation of close to zero when looking at the 1st and the 2nd principal components. On the other hand, for genuine and counterfeit notes, the same attributes have a negative correlation with respect to the 1st and the 2nd principal component. For counterfeit notes, left, right and length hardly have any variation since they are low on loadings for the first 2 principal components. In case of genuine notes, however, they almost contribute equally and are correlated. For counterfeit notes the right left and length attributes are positively correlated whereas for genuine notes they have a correlation of almost -1. The loadings for all the three cases are shown in figures 7, 8 and 9.

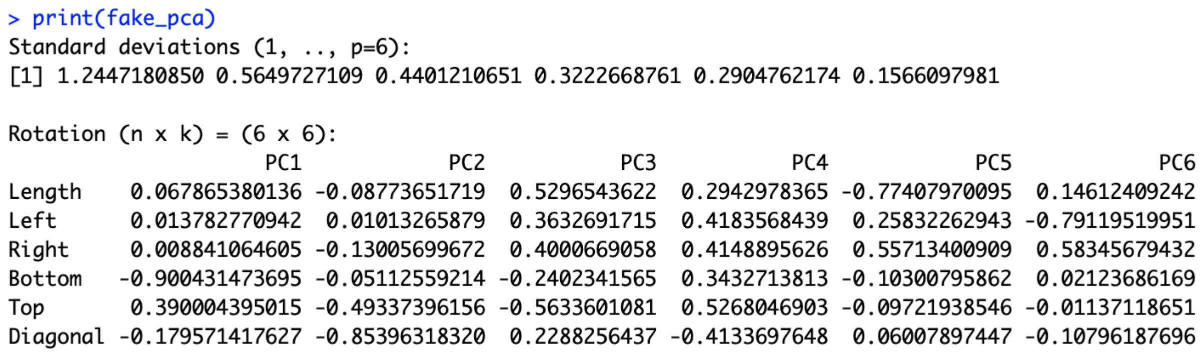
Diagnostic plots are shown in figures 10, 11, 12 and 13.



```
> print(gen_pca)
Standard deviations (1, .., p=6):
[1] 0.8300895575 0.5993958663 0.4308224301 0.2953559635 0.2831666938 0.2047786183

Rotation (n x k) = (6 x 6):
                  PC1           PC2            PC3           PC4           PC5            PC6
Length    0.06138959157 -0.3785417196  0.47131651136  0.7860846654 -0.1128435402 -0.01163893790
Left      0.01270596582 -0.5065530139  0.10146241995 -0.2431857277  0.3597949730  0.73780502785
Right     0.03734416682 -0.4542917131  0.19626283776 -0.2802038626  0.4807942233 -0.66635430619
Bottom    0.69695839924 -0.3577669959 -0.10770325940 -0.2431907290 -0.5594590085 -0.05015636982
Top      -0.70545820834 -0.3648023898  0.07370965402 -0.2443523108 -0.5479895967 -0.06175896427
Diagonal  0.10608013813  0.3642131836  0.84380917084 -0.3543552082 -0.1156021078  0.07176617248
```

Figure 7: Loadings of the principal components for genuine notes



```
> print(fake_pca)
Standard deviations (1, .., p=6):
[1] 1.2447180850 0.5649727109 0.4401210651 0.3222668761 0.2904762174 0.1566097981

Rotation (n x k) = (6 x 6):
                   PC1            PC2           PC3           PC4            PC5            PC6
Length    0.067865380136 -0.08773651719  0.5296543622  0.2942978365 -0.77407970095  0.14612409242
Left      0.013782770942  0.01013265879  0.3632691715  0.4183568439  0.25832262943 -0.79119519951
Right     0.008841064605 -0.13005699672  0.4000669058  0.4148895626  0.55713400909  0.58345679432
Bottom   -0.900431473695 -0.05112559214 -0.2402341565  0.3432713813 -0.10300795862  0.02123686169
Top       0.390004395015 -0.49337396156 -0.5633601081  0.5268046903 -0.09721938546 -0.01137118651
Diagonal -0.179571417627 -0.85396318320  0.2288256437 -0.4133697648  0.06007897447 -0.10796187696
```

Figure 8: Loadings of the principal components for counterfeit notes

```
> print(all_pca)
Standard deviations (1, .., p=6):
[1] 1.7321388139 0.9672747917 0.4933697461 0.4412014783 0.2919106904 0.1884533797

Rotation (n x k) = (6 x 6):
                  PC1            PC2          PC3           PC4            PC5            PC6
Length    0.04377427189 -0.01070966117 0.3263164974 -0.5616917677 -0.75257277768  0.09809807413
Left     -0.11216158701 -0.07144696645 0.2589613665 -0.4554587919  0.34680081517 -0.76651197219
Right    -0.13919062370 -0.06628207600 0.3447327417 -0.4153296258  0.53465173358  0.63169677788
Bottom   -0.76830498683  0.56307224704 0.2180221980  0.1861082220 -0.09996770862 -0.02221710868
Top      -0.20176610266 -0.65928987979 0.5566856769  0.4506985058 -0.10190228881 -0.03485874054
Diagonal  0.57890193368  0.48854254632 0.5917628489  0.2584483225  0.08445894526 -0.04567946417
```

Figure 9: Loadings of the principal components for all notes

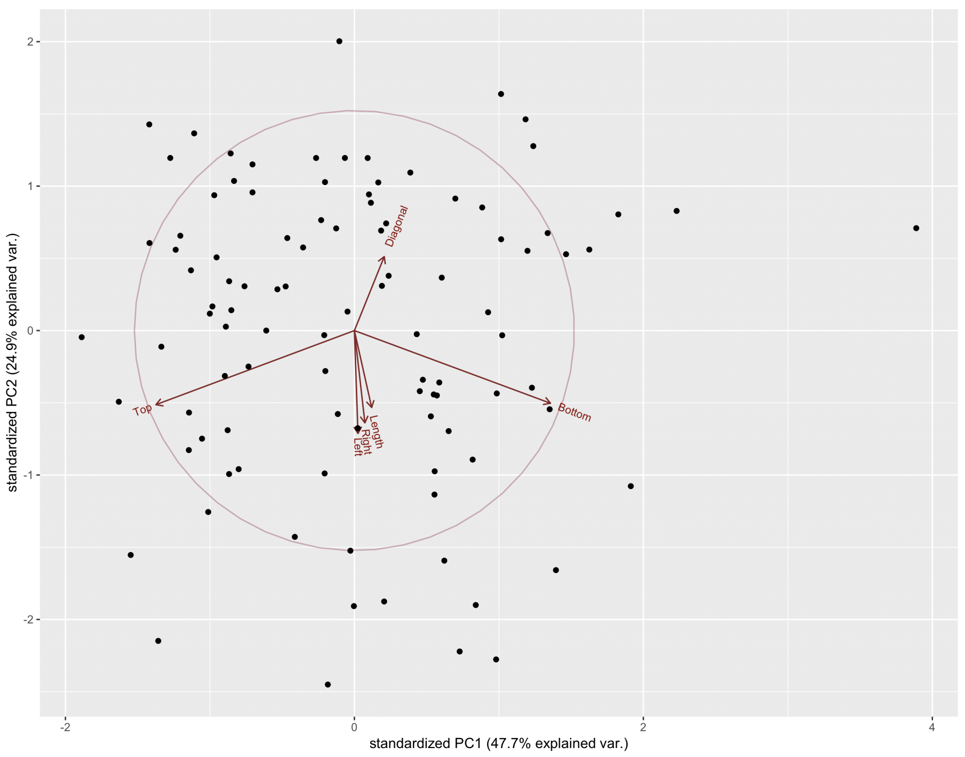


Figure 10: Percentage variance explained by PC1 and PC2 for genuine notes

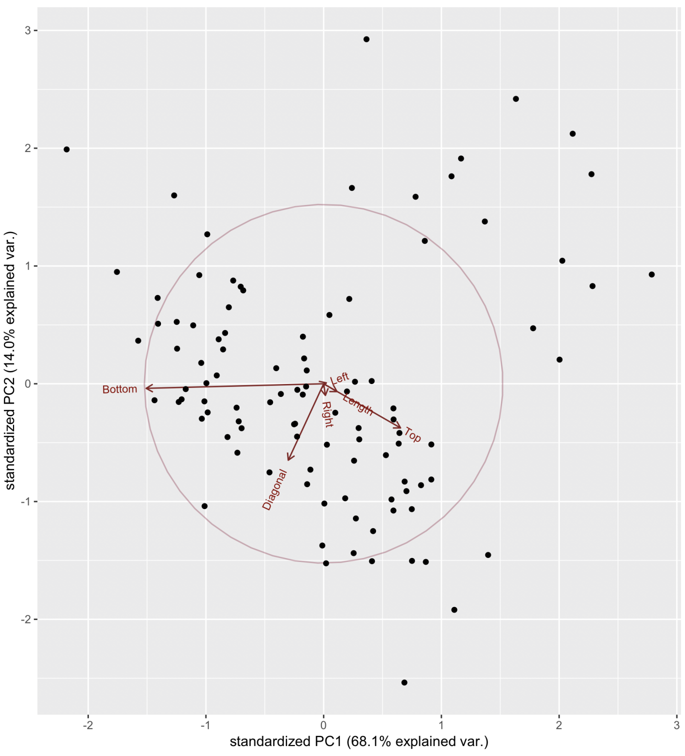


Figure 11: Percentage variance explained by PC1 and PC2 for counterfeit notes

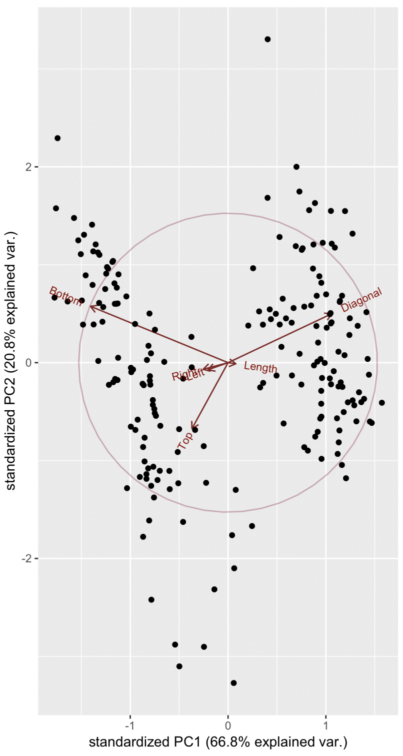


Figure 12: Percentage variance explained by PC1 and PC2 for all notes

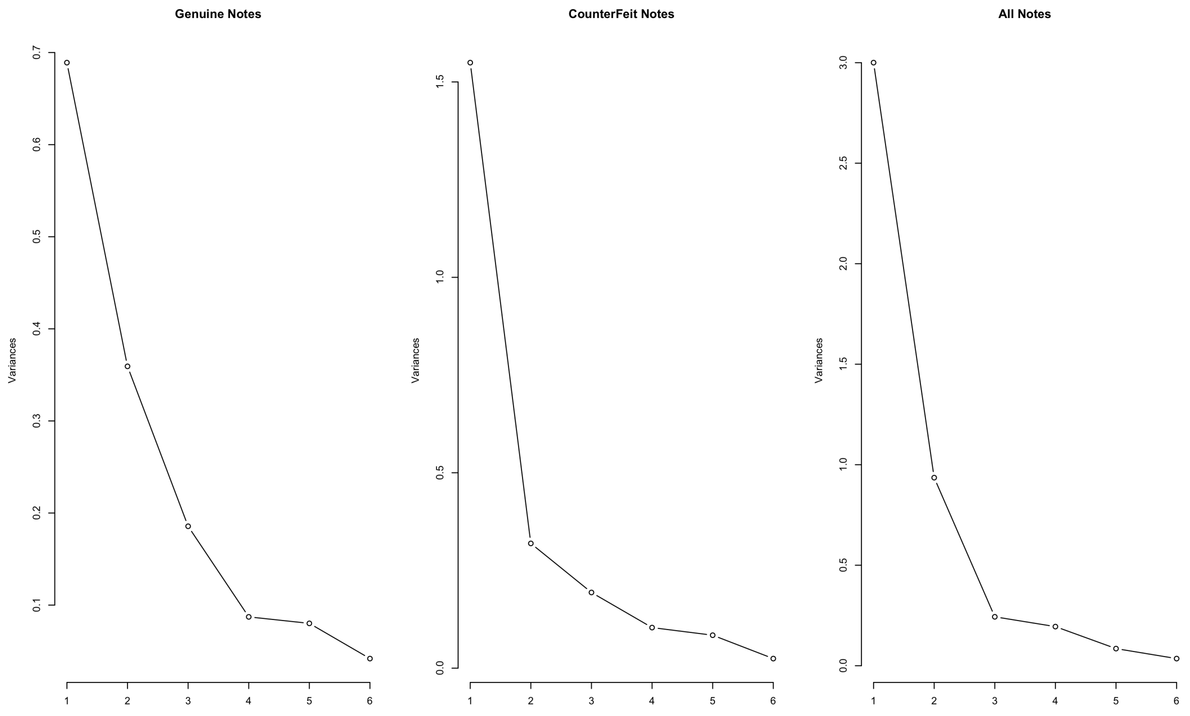


Figure 13: Generalization (Test Set) error vs Number of neurons in the hidden layer