University College Dublin

UCD Michael Smurfit Graduate Business School
MSc Business Analytics
MIS41120: Statistical Learning

Practical Data Analysis Assignment

Hang Nguyen **Student ID 19202198**

Pengwei Song **Student ID 19203158**

Yijun Wang **Student ID 19201028**

Please note that plagiarism in any form is forbidden in all examinations, theses or other academic exercises.
We confirm that this paper is our own unaided work and all sources are cited.

*Hang Nguyen :*____Hang__________
*Pengwei Song :*____Pengwei_______
*Yijun Wang :*____Yijun_________

DATE SUBMITTED: Jul 28th, 2020 TIME: 11.00 PM

Contents

List of Figures

List of Tables

1 Introduction

In this report, we adopt three statistical techniques, which are regression, SVM, and MlP to create models and analyse two datasets, Boston and Default of Credit Card Clients. Respectively, we carry out three regularisation approaches to regularise these models. Accordingly, we apply k-fold cross-validation to estimate the model quality.

The report contains five main sections: Introduction, Datasets, Methods, Results, and Conclusion.

Section 1 recaps the background of the analysis, and then details how this report is structured.

Section 2 describes the two datasets we adopted to conduct the following analysis.

Section 3 details the methods we applied to build models

Section 4 elaborates on the results provided by each model

Section 5 summarises the findings and suggests the recommendation.

Appendixes regarding model details and a reference list is also included in the report.

2 Data sets

Boston data set
According to [1], the Boston data set is derived from information collected by the U.S. Census Service concerning housing in the area of Boston Mass. The data set contains a total of 506 cases and 14 attributes (output variable: crim). The attributes are shown in the Table 2 in the Appendix A.

Default Of Credit Card Clients data set
According to [2], the Default data set is derived from information collected by the Department of Information Management, Chung Hua University, Taiwan. This data set contains information on default payments, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005. The data set contains a total of 30,000 cases and 25 attributes (output variable: default.payment.next.month). The attributes are shown in Table 3 in Appendix A.
There are four reasons for us to choose this data set. Firstly, this data set provides us with an opportunity to apply all three models on a classification problem to test the different regularisation approaches. Secondly, the data set is rich enough with more than 20 predictors, thousands of instances and, yet still lightweight so it could be easily downloaded. Thirdly, the data does not require much cleaning (no categorical variable, no missing data ...), except for balancing data, so the authors could focus on training and evaluating the models. Finally, this is a simple binary classification problem so the running time would not be too long.

The data set is available at `https://archive.ics.uci.edu/ml/machine-learning-databases/00350/default%20of%20credit%20card%20clients.xls`

3 Methods

3.1 Regression models

Overview
The methods we apply to analyse the two datasets, Boston and Default, are multiple linear regression and logistic regression respectively.
Multiple linear regression is similar to simple linear regression , but with two or more predictors (x1, x2, x3... etc.). It is a widely adopted statistical technique with high interpretability that analyse the predictors' effect on response (y).
Logistic regression will be applied to determine categorical response variables (e.g. "yes" or "no", "1" or "0"). To predict which class a value belongs, we have to set a threshold, and based on which the obtained estimated probability is classified into classes.

Regularisation
In the first analysis, we delve in three approaches, which are ridge regression (l2), lasso (l1), and elastic net (l1 and l2) respectively to regularise the linear model. To be more specific, the dataset Boston is split into training set and testing set for all the regularization approaches; then we apply the training set to build a model, the quality of which is estimated by using a 10-fold cross-validation to determine the best lambda, which results in the lowest training MSE. Additionally, in the elastic net approach, we also determine the best alpha, which is default setting as 0 and 1 in ridge regression and lasso. Once the values of lambda and alpha are determined, the model is used to predict the response for the testing dataset, getting the test MSE.
In the second analysis, we also use ridge regression (l2), lasso (l1), and elastic net (l1 and l2) to regularise the logistic regression. Specifically, we randomly split the Default dataset, which contains 30000 instances, into training set (24000 instances) and testing set (6000 instances). Then we conduct the same procedure as we did to Boston dataset to determine the best lambda and alpha, predicting the response for the testing dataset and calculating the test accuracy and AUC. One more to mention, in the Default analysis, we set a threshold of 0.5, if the response value y 0.5, then we classify the case as default else as not default.

3.2 Support Vector Machine model (SVM)

Overview
SVM methodology is a supervised learning method that can be used for classification. It is especially useful for two classes prediction. The main idea of SVM is to transform the input pattern space from low dimension to higher dimension so that the input variables which is not linearly separable in low dimension can be well split by linear hyperplane in high dimension space. Considering the computational cost of transforming inputs into high dimension, kernels are used to implicitly map input patterns.

Our aim is to maximize the margin of hyperplane and let it separates the classes as far as possible. We may allow misclassification in SVM to avoid overfitting. In this report, We use l1 norm, l2 norm and elastic net(l1l2) to regularize svm model.

3.3 MultiLayer Perceptron model (MLP)

Overview
A multilayer perceptron (MLP) is a class of feedforward artificial neural network. It is a supervised learning algorithm that learns a function $f(.) : R^m \longrightarrow R^o$ by training on a dataset, where m is the number of dimensions for input and o is the number of dimensions for output ([3]. Figure 12 in Appendix D show a one hidden layer MLP with scalar output.

Data Preprocessing
Firstly, the authors check for missing data, omit the ID column, and change the target feature into $default$. Secondly, the authors split the raw data into train sets X_train and Y_train (80%) and test sets X_test and Y_test (80%). Thirdly, because in both data sets, the features have different ranges so we normalise data into $0-1$ range. The output $scaled_{Y}hat$ for the test Boston data set are rescaled back to the original scale as Y_hat. Finally, as the Default data set is imbalanced, we set the class weights based on the ratio of positive/negative outcome in the $default$ feature.

Neural network construction
A simple neural network architecture could be construct in Keras by using sequential function, which is built by stacking layers sequentially. We are stacking two hidden layers for both Boston data set and Default data set, and all are dense layers (layers with fully connected nodes, as illustrated below in Figure 13). Summaries of architecture are presented in the Appendix D.

Traditionally, the weights and the bias of a neural network were set to small random numbers. However, we could boost the model performance by initialise the weights with different distributions and initialise the biases with small positive values instead of *zeros* (by default). Consulted from [4], in this project, the authors do not adjust the bias initialisation but incorporate the *orthogonal* approach for weights initialisation. The weight initialisation approach comparison in terms of step loss is shown in Figure 16 in Appendix D. This method is also proved in [5] that it could avoid explosion or vanishing effects (activations and gradients could grow big or small quickly [6]).

In a neural network, the activation function is responsible for transforming the summed weighted input from the node into the activation of the node or output for that input. For hidden layers, the author are using *ReLU* (the Rectified Linear Unit activation function). It is a piecewise linear function ($f(x) = max(0, x)$)that will output the input directly if is positive, otherwise, it will output zero [7]. For output layers, we are using *Linear* activation function ($f(x) = ax$) for Regression problem and *Sigmoid* activation function ($f(x) = 1/(1 + e^{-x})$)) for Classification problem [8]. All three activation functions are illustrated in Figure 17, 18 and 19 in Appendix D.

After setting up the starting point (initialiser), the authors proceed to choose the optimizer, tune the learning rate, and the batch size. We choose the *Adam* optimisation approach - an extension to stochastic gradient descent. Empirical results demonstrate that *Adam* works well in practice and compares favorably to other stochastic optimization methods [9], partly because it uses the most memory for a given batch size [10]. After tuning, we choose the batch size of 1 (for Boston data set) and 32 (for Default data set), and the learning rate of 0.00001. Then for train/validation history over 100 epochs, we set the metric as MSE (loss) for Regression problem, and accuracy for Classification problem. Besides, some of the other useful techniques for neural network are dropout regularisation, callbacks and early stopping could be incorporate in building model.

K-Fold Cross Validation
K-Fold Cross Validation is recommended as one of the best method for model validation. The 5-Fold cross validation is set up for train data set (80% of raw data). For the Regression problem, we evaluate 4 models (one baseline model and three models using regularisation approaches) then calculate the Average normalised MSE for each model from normalised MSE on 5 validation date sets.

Regularisations
To avoid overfitting, 3 regularization approaches from built-in function in Keras are experimented, creating 3 more MLP model, which are MLP L2 (ridge regression), MLP L1 (lasso) and MLP L1_L2 (elastic net) models, besides the MLP (baseline) model.

Model Comparison

For the Regression problem, based on the MSE history analysis and the Average MSE on the validation data set, the authors pick the best model that has the lowest MSE. All of them are normalised MSE, and the real MSE is calculated after inverting the $\hat{y}$ back to the original scale.
For the Classification problem, the model optimality is assessed by comparing among accuracy histories analysis and the Average accuracy on the validation data set. The authors do not rescale the classification output, as the activation function for the output is set as sigmoid function.

4 Results

4.1 Regression models

Regression problem
Based on the Boston dataset, we first construct the initial model and then regularise it with three regularisation methods.
We explore the initial multiple linear model to complete the part (b) of Question 15 in the textbook ISLR. As we can see from Figure 1 , the p values of predictors, such as zn, dis, rad, black, and medv are less than 0.05, indicating that these predictors are significant, and for which we can reject the null hypothesis H0: j= 0. For all other predictors, there is no sufficient evidence to reject the null hypothesis that their coefficients are 0.
Figure 2 displays the coefficient of each predictor alone with the test MSE and the best lambda. As indicated, the values of test MSE are quite close, and lasso gives a better result with the lowest test MSE (47.959691), closely followed by ridge regression (48.197345) and elastic net (48.246552).

Classification problem
Similarly, based on the Default dataset, we first create the initial model, which, then, is regularised by the regularization approaches.
Unlike the numerical dependent variable in Boston dataset, the dependent variable in Default dataset is categorical. As a result, we generate a logistic regression model to analyse the dataset. From Figure 3 , we can find that predictors like LIMIT_BAL, SEX, EDUCATION, MARRIAGE, AGE, PAY_0, PAY_2, PAY_3, BALL_AMT1, PAY_ATM1, PAY_ATM2, and PAY_ATM6 are significant. Besides, the model gives an accuracy of 0.812333 and AUC of 0.725055.
The results of each model are given in Figure 4, where we can find that elastic net has the highest accuracy (0.811117) and AUC (0.724267). The accuracy and AUC of lasso are 0.810833 and 0.724052; that of ridge regression are 0.806667 and 0.723005 respectively.

4.2 Support Vector Machine model (SVM)

Implement SVM on Boston dataset
Since SVM performs on binary classification, we need to convert the response "crim" to categorical input. After looking through the response column using hist(), we subjectively convert the crime rate under 1.0 to class "-1" and greater than 1.0 to class "1".

Apply l2 norm on Boston data
We set 70 percent (354) of overall data as training and 30 percent (152) as testing. First we implement svm with basic l2 norm on Boston data. The optimal cost = 10 and gamma = 0.01. Using that model to predict testing data, we found that it gives us very good predicting model with high accuracy rate 0.95 and sensitivity is 0.88.

The black curve shown in Figure 5 shows the ROC with optimal parameter and red curve with the other parameter.The black curve's position is more to the top-left corner, which represents that the model has a better prediction on both training and testing data.

Apply l1 norm on Boston data
We set 10 lambda values which range from 0.01 to 100 with equal interval and let the function to decide which one is the best through 10-cross validation. The function svmfs() gives optimal lambda = 100 and the accuracy rate is 0.95 on testing dataset.

Apply Elastic net(l1l2) norm on Boston data
We subjectively set alpha = 0.5 at first so that lasso and ridge regression have the same impact on penalty. The minimum lambda is 0.38 after going through 10-fold cross validation in Figure 6. 9 variables are selected as input variables. The accuracy is 0.86 on testing data.

If we set alpha = 0.7, the optimal lambda will be 0.27 and the accuracy decreases to 0.72. Under this condition, only 8 variables are selected. If alpha = 0.3, the optimal lambda value will be 0.56 and accuracy rate is 0.85 with 11 variables selected. The reason for that is because alpha=1 is the lasso penalty and alpha=0 the ridge penalty. If the alpha is closer to 1, then the model will perform more likely as lasso. Since lasso has the good ability to shrinkage the coefficients to 0, it explains why we have less inputs as alpha increase.

Implement SVM on Clients default dataset

"Clients default dataset" is a binary classification problem which perfectly fits svm methodology. So the only thing we need to do is converting the class label "0" to "-1" by ifelse() to fit the package requirements.
As the Figure 7 shows in Appendix C, 385 clients will not default next month, and 115 clients will default. There is an imbalanced class in our dataset. It warns us to be careful about accuracy paradox of the model we build.

Apply l2 norm on "Default" dataset

In the SVM part, we set 70 percent of overall data as training and 30 percent as testing. Considering the computation time of SVM is very long for 30000 instances, we took a sample of 1000 of dataset. In order to make l2 norm of svm, we use svm() in e1071 package using radial kernel with random choosing gamma = 20 and cost = 100. The initial parameter are chosen randomly to have a first look. The model name is "out".

From the Figure 8, 696 instances are used as support vectors, from which 147 are default class and 549 are not default class. We further explored the confusion matrix of the model and found that the "out" model gives a perfect split on training dataset without any misclassification. However, sometimes the model will overfit the training data, so the performance on testing data is more important.
From the figure above, we found that the model "out" with radial kernel preforms poorly on testing data since it predicts all the default class as non-default. The reason for that is because of the imbalanced class in the dataset. Thought the accuracy rate 0.78 is high, the model fell into the accuracy paradox.
In order to find the optimal parameter of SVM, we create a list of range of parameters and use tune() to find out the optimal value. We use 10-fold cross validation to verify the result.

The validation process in Figure 9 gives us the optimal cost value = 10 and gamma = 0.01. This pair of parameter gives us the least error rate 0.19 compare to the other values. The accuracy rate of the model with optimal parameter is 0.77. The "bestmod" gives an improvement on predicting clients who will default next month. The true positive rate is improved from 0 to 0.16.
We also try linear kernel with cost = 0.01, 0.10, 1.00, and 10.0 to see if the linear kernel gives a better prediction on default clients.The result shows that the best cost parameter is 1.0 with ac-

curacy rate 0.78 and sensitivity rate 0.13, which does no give a huge improvement from radial kernel.

Considering time efficiency in Figure 10, we used system.time() to calculate the computation time of svm with l2 norm. The "elapsed" in Figure shows the wall clock time taken to execute the function. After running the function multiple times, we found that SVM with radial kernel takes 0.08s and linear kernel takes 0.06s on average.

Apply l1 norm on "Default" dataset
We used package "penalizedSVM" to implement l1 norm of svm on training dataset. Because functions in "penalizedSVM" cost too much time to run, we take a sample of 500 instance for l1 norm. Since we found that the number of non-default is three times more than the number of default in overall dataset, we set class weight "-1": 1 and "1": 3. The lambda range = c(0.1,1,10,100) to find out the optimal lambda value. We set maximum iteration = 700 with 5-fold cross validation.

The running result gives the best lambda value = 10. 19 input variables are selected and the corresponding coefficients are given in the output window. The bias of the model is -0.38. The accuracy rate of the model "norm1.fix" on testing dataset is 0.70 with sensitivity rate 0.23. Comparing to the model with l2 norm, the model with l1 norm has a better prediction on default clients. The computation time of svm with l1 norm is 24.05s, which takes 300 times longer than the basic l2 norm.

Apply Elastic net(l1l2) norm on "Default" dataset
We use R package sparseSVM to implement elastic net norm. Initially the alpha = 0.5 means that lasso and ridge penalty contribute the same proportion in the model.

The Figure 11 shows that optimal lambda equals to 0.03 under 5-fold validation. 20 variables are selected as the input of the model. The accuracy rate is 0.70. With smaller alpha value 0.3, the model has more selected inputs 23 and slightly less accuracy rate 0.69. If we input larger alpha value = 0.8, lasso(l1) norm will contribute most of the penalty to the model. The number of input and accuracy rate is the same as 20 and 0.70.

4.3 MultiLayer Perceptron model (MLP)

Regression problem
The histories of train/validation loss on Boston data set are illustrated in Figure 21, 22, 23 and 24. Looking at these histories, we could clearly see that MLP model (baseline model) outperforms the others on both the train and validation sets, followed by MLP L2 (ridge regression) model, MLP L1_L2 model (elastic net) and L1 model (lasso), respectively. The average normalised MSE of MLP models on Boston validation data set, shown in Figure 25, observed the same pattern. That means the regularisations have not much effect on the data set. Additionally, the validation loss is either stable or slight decrease throughout 100 epochs, and is always less than the training loss. It shows that the MLP model for the Boston data set is stable and has good performance.
The final MSE and R squared of MLP model on Boston test data set are shown in Figure 26. The low R squared reveals the low accuracy of model prediction. The MSE of MLP best model is higher than the MSE of Linear regression best model. It proves that for small data sets, Linear regression is better than Neural network, specifically MLP in this case. Otherwise, the hidden layers of MLP are difficult to interpret. Finally, the MLP, and Neural Network models in general, could be computationally expensive if not run in a GPU/TPU.

Classification problem
The histories of train/validation accuracy on Default data set are illustrated in Figure 27, 28, 29 and 30. Looking at these histories, we could clearly see that MLP L1_L2 model (elastic net) outperformed the others on both the train and validation sets, followed by MLP L1 (lasso) model, MLP

L2 model (ridge regression) and MLP baseline model, respectively. The average accuracy of MLP models on Default validation data set, shown in Figure 31, observed the same pattern. However, because the accuracy alone is not sufficient for model assessment, we apply these models on test set to see their confusion matrices, and then calculate the precision and recall. The final confusion matrices of MLP models on Default test data set are shown in Figure 32 and 33, and the AUC values in Figure 35. Also, the summary of MLP models' result and actual result on Default test data set is presented in Figure 34. The Accuracy, Precision, Recall and AUC values on the test data sets of 4 models are presented in the Table 1 as follows.

Metrics	**MLP**	**MLP L2**	**MLP L1**	**MLP L1_L2**
Accuracy	0.773	0.752	0.773	0.775
Precision	0.500	0.461	0.501	0.505
Recall	0.530	0.542	0.536	0.519
AUC	0.687	0.678	0.689	0.685

Table 1: Assessment of the MLP models on Default test data set

Table 1 above reveals that although MLP L1_L2 model has a highest accuracy and precision, it has the second lowest AUC score on test set. This is because for imbalanced classification with a severe skew and few examples of the minority class like Default data set, a small number of correct or incorrect predictions can result in a large change in AUC score [11]. In terms of accuracy, MLPs do not give the accuracy as high as both Logistic Regression and SVM. It is partly due to the fact that hyperparameter tuning for Neural Network is expensive so that the authors could not incorporate the tuning process into the K-Fold cross validation.

5 Conclusion

Through implementing regression, SVM and MLP with l1, l2 and elastic net regularisation on two dataset with different property, we found that each statistic learning method has its own advantage on a certain type of data, and vice-versa, may perform poorly on some other patterns of data.

For Boston data, regression model with l1 norm performs best among all three methodology, following by MLP and SVM. MLP is better used on large dataset but the Boston data only has hundreds of rows and limited features. SVM do have a high accuracy rate, but it most suited for binary classification. We subjectively set the threshold and convert the data into a classification one. The threshold needs to be further validated.

For clients default data, which the response is binary class, SVM with l2 norm gives the highest accuracy and sensitivity, following with MLP and regression model. The MLP has almost the same accuracy as SVM, which demonstrates the ability of neural network method on large dataset. However, regression model which does great job on Boston data has a bad performance on clients default data.

References

[1] "The boston housing dataset," *Department of Computer Science, University of Toronto*, 1996.

[2] U. M. L. Repository, "default of credit card clients data set," *Center for Machine Learning and Intelligent Systems*, 2016.

[3] "Neural network models (supervised)," *scikit-learn developers*.

[4] A. Hosny, "Priming neural networks with an appropriate initializer," *becominghuman.ai*, 2017.

[5] D. Mishkin and J. Matas, "All you need is a good init," *Center for Machine Perception, Czech Technical University in Prague*, 2015.

[6] K. . Kunin, "Initializing neural networks," *deeplearning.ai*, 2019.

[7] J. Brownlee, "A gentle introduction to the rectified linear unit (relu)," *Machine Learning Mastery*, 2019.

[8] D. Gupta, "Fundamentals of deep learning – activation functions and when to use them?," *Analytics Vidhya*, 2020.

[9] J. Brownlee, "Gentle introduction to the adam optimization algorithm for deep learning," *Machine Learning Mastery*, 2019.

[10] K. K. et al., "Parameter optimization in neural networks," *deeplearning.ai*, 2019.

[11] J. Brownlee, "Roc curves and precision-recall curves for imbalanced classification," *Machine Learning Mastery*, 2019.

A Appendix: Data sets

Attributes	**Description**
crim	per capita crime rate by town
zn	proportion of residential land zoned for lots over 25,000 sq.ft.
indus	proportion of non-retail business acres per town.
chas	Charles River dummy variable (1 if tract bounds river; 0 otherwise)
nox	nitric oxides concentration (parts per 10 million)
rm	average number of rooms per dwelling
age	proportion of owner-occupied units built prior to 1940
dis	weighted distances to five Boston employment centres
rad	index of accessibility to radial highways
tax	full-value property-tax rate per $10,000
ptratio	pupil-teacher ratio by town
black	$1000(\text{Bk} - 0.63)^2$ where Bk is the proportion of blacks by town
lstat	lower status of the population (percent)
mdev	median value of owner-occupied homes in $1000s

Table 2: Variable Descriptions of the Boston Data Set

Attributes	**Description**
ID	ID of each client
LIMIT_BAL	Amount of given credit in NT dollars (includes individual and family/supplementary credit
SEX	Gender (1=male, 2=female)
EDUCATION	1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)
MARRIAGE	Marital status (1=married, 2=single, 3=others)
AGE	Age in years
PAY_0	Repayment status in September, 2005 (-1=pay duly, 1=payment delay for one month, 2=payment delay for two months, ... 8=payment delay for eight months, 9=payment delay for nine months and above)
PAY_2	Repayment status in August, 2005 (scale same as above)
PAY_3	Repayment status in July, 2005 (scale same as above)
PAY_4	Repayment status in June, 2005 (scale same as above)
PAY_5	Repayment status in May, 2005 (scale same as above)
PAY_6	Repayment status in April, 2005 (scale same as above)
BILL_AMT1	Amount of bill statement in September, 2005 (NT dollar)
BILL_AMT2	Amount of bill statement in August, 2005 (NT dollar)
BILL_AMT3	Amount of bill statement in July, 2005 (NT dollar)
BILL_AMT4	Amount of bill statement in June, 2005 (NT dollar)
BILL_AMT5	Amount of bill statement in May, 2005 (NT dollar)
BILL_AMT6	Amount of bill statement in April, 2005 (NT dollar)
PAY_AMT1	Amount of previous payment in September, 2005 (NT dollar)
PAY_AMT2	Amount of previous payment in August, 2005 (NT dollar) $1000s
PAY_AMT3	Amount of previous payment in July, 2005 (NT dollar) $1000s
PAY_AMT4	Amount of previous payment in June, 2005 (NT dollar)
PAY_AMT5	Amount of previous payment in May, 2005 (NT dollar)
PAY_AMT6	Amount of previous payment in April, 2005 (NT dollar)
default.payment.next.month	Default payment (1=yes, 0=no)

Table 3: Variable Descriptions of the Default Data Set

B Appendix: Regression Models

```
Call:
lm(formula = crim ~ ., data = Boston[train, ])

Residuals:
   Min     1Q Median     3Q    Max
-8.259 -2.211 -0.171  1.199 52.405

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 15.840477   8.392398   1.887   0.0601 .
zn           0.044897   0.020964   2.142   0.0331 *
indus       -0.102836   0.101461  -1.014   0.3117
chas        -0.050220   1.407742  -0.036   0.9716
nox         -9.061665   6.686489  -1.355   0.1765
rm          -0.291825   0.782961  -0.373   0.7096
age          0.013859   0.021003   0.660   0.5099
dis         -0.904479   0.332291  -2.722   0.0069 **
rad          0.578090   0.106992   5.403 1.42e-07 ***
tax         -0.002097   0.006284  -0.334   0.7389
ptratio     -0.267021   0.213152  -1.253   0.2114
black       -0.001293   0.004088  -0.316   0.7521
lstat        0.154273   0.085691   1.800   0.0729 .
medv        -0.150532   0.072206  -2.085   0.0380 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.709 on 275 degrees of freedom
  (17711 observations deleted due to missingness)
Multiple R-squared:  0.5421,    Adjusted R-squared:  0.5204
F-statistic: 25.04 on 13 and 275 DF,  p-value: < 2.2e-16
```

Figure 1: Summary of Boston Crime Model

	RidgeRegression	Lasso	ElasticNet
Test MSE	48.1973449	47.9596907	48.2465522
BestLambda	0.5542937	0.0364688	0.4908443
X.Intercept.	8.9623751	13.8222574	8.3038806
zn	0.0327204	0.0381520	0.0317927
indus	-0.0811390	-0.0724674	-0.0679225
chas	-0.7368191	-0.6176489	-0.6775196
nox	-5.3428681	-7.9624473	-4.1403607
rm	0.3374047	0.2922722	0.2432881
age	0.0018803	0.0000000	0.0000000
dis	-0.6970623	-0.8470134	-0.6499666
rad	0.4206830	0.5271866	0.4334929
tax	0.0034682	-0.0004990	0.0023717
ptratio	-0.1346336	-0.2127891	-0.1059533
black	-0.0084889	-0.0075428	-0.0083321
lstat	0.1426571	0.1263267	0.1376434

Figure 2: Test MSE

```
Call:
glm(formula = default ~ ., family = binomial, data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.1249  -0.7011  -0.5481  -0.2906   3.2104

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -6.762e-01  1.323e-01  -5.111 3.20e-07 ***
LIMIT_BAL   -8.169e-07  1.761e-07  -4.638 3.52e-06 ***
SEX         -9.909e-02  3.430e-02  -2.889 0.003868 **
EDUCATION   -1.048e-01  2.356e-02  -4.448 8.68e-06 ***
MARRIAGE    -1.556e-01  3.533e-02  -4.403 1.07e-05 ***
AGE          7.243e-03  1.988e-03   3.643 0.000269 ***
PAY_0        5.663e-01  1.980e-02  28.601  < 2e-16 ***
PAY_2        7.982e-02  2.259e-02   3.533 0.000411 ***
PAY_3        6.848e-02  2.524e-02   2.713 0.006669 **
PAY_4        2.901e-02  2.792e-02   1.039 0.298795
PAY_5        2.467e-02  2.988e-02   0.825 0.409091
PAY_6        1.900e-02  2.461e-02   0.772 0.440271
BILL_AMT1   -4.691e-06  1.231e-06  -3.812 0.000138 ***
BILL_AMT2    1.951e-06  1.647e-06   1.184 0.236362
BILL_AMT3    1.360e-06  1.464e-06   0.928 0.353220
BILL_AMT4   -2.972e-07  1.489e-06  -0.200 0.841833
BILL_AMT5   -5.530e-07  1.769e-06  -0.313 0.754562
BILL_AMT6    1.349e-06  1.419e-06   0.950 0.341902
PAY_AMT1    -1.576e-05  2.754e-06  -5.722 1.05e-08 ***
PAY_AMT2    -9.031e-06  2.320e-06  -3.892 9.93e-05 ***
PAY_AMT3    -2.488e-06  1.951e-06  -1.275 0.202278
PAY_AMT4    -3.497e-06  1.954e-06  -1.789 0.073549 .
PAY_AMT5    -3.061e-06  1.926e-06  -1.590 0.111906
PAY_AMT6    -3.666e-06  1.611e-06  -2.275 0.022890 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 25365  on 23999  degrees of freedom
Residual deviance: 22341  on 23976  degrees of freedom
AIC: 22389

Number of Fisher Scoring iterations: 6
```

Figure 3: Summary of Default Model

```
              model accuracy      auc
1: RidgeRegression 0.806667 0.723005
2:           Lasso 0.810833 0.724052
3:      ElasticNet 0.811117 0.724267
```

Figure 4: Test Accuracy and AUC

C Appendix: Support Vector Machines Models

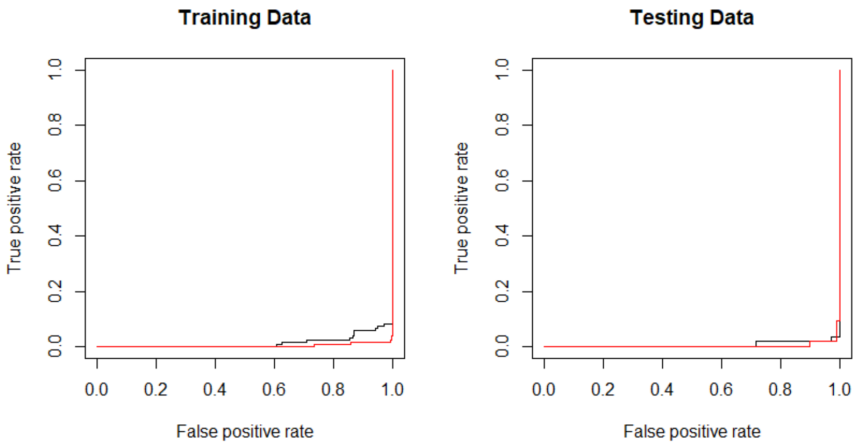

Figure 5: ROC curve of svm on Boston data

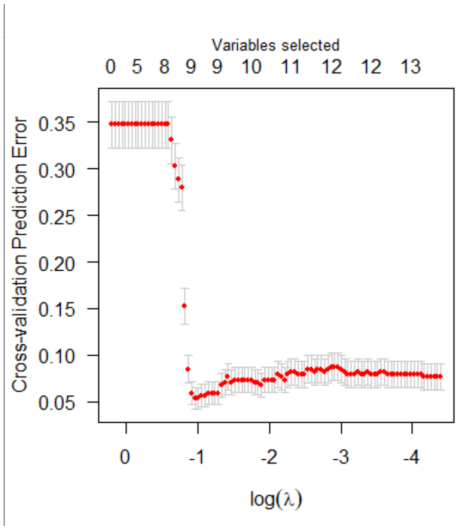

Figure 6: 10-fold cross validation prediction error on Boston data

```
> table(data$y)

 -1   1
786 214
```

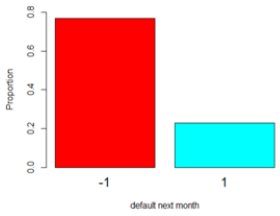

Figure 7: Imbalanced class in dataset

```
> summary(out)

Call:
svm(formula = y ~ ., data = datTrain, kernel = "radial",
    gamma = 20, cost = 100)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  100

Number of Support Vectors:  696

 ( 147 549 )


Number of Classes:  2

Levels:
 -1 1
```

```
> #Confusion matrix on training data
> table(out$fitted,datTrain$y)

     -1   1
  -1 549   0
  1    0 147
```

Figure 8: Results of l2 norm of svm on default data

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
   10  0.01

- best performance: 0.1898137

- Detailed performance results:
    cost gamma     error dispersion
1  1e-01 1e-02 0.2113251 0.03340962
2  1e+00 1e-02 0.2112836 0.02858664
3  1e+01 1e-02 0.1898137 0.04934247
4  1e+02 1e-02 0.1912836 0.05487315
5  1e+03 1e-02 0.2314700 0.05616599
6  1e-01 1e-01 0.2113251 0.03340962
7  1e+00 1e-01 0.1983644 0.03807741
8  1e+01 1e-01 0.2313458 0.03731219
9  1e+02 1e-01 0.2601863 0.04818777
10 1e+03 1e-01 0.2831056 0.04616108
```

Figure 9: Cross validation l1 svm on Default data

```
> system.time(svm(y~.,data=datTrain,kernel="radial",
+                 gamma=0.01, cost=10))
   user  system elapsed
    0.1     0.0     0.1
> system.time(svm(y~.,data=datTrain,kernel="linear",
+                 cost=1))
   user  system elapsed
   0.06    0.00    0.06
```

Figure 10: Computation time of svm l2 norm on default data

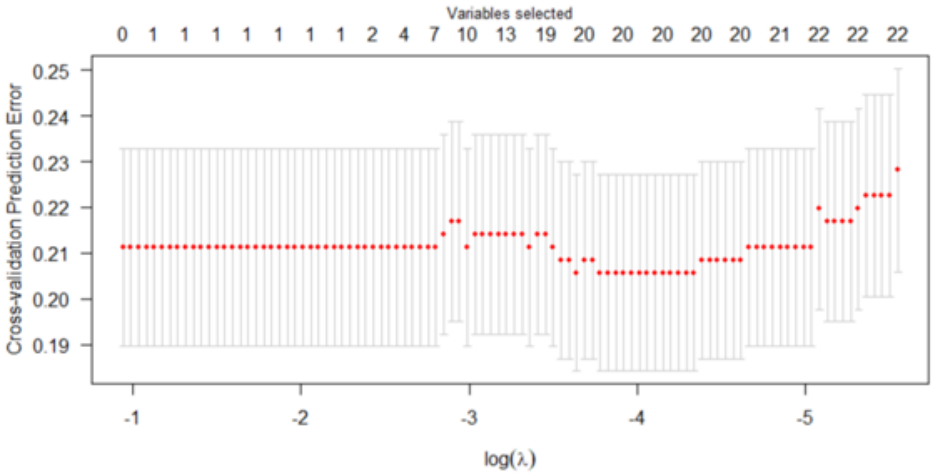

Figure 11: Cross validation on default data

D Appendix: Neural Network Models

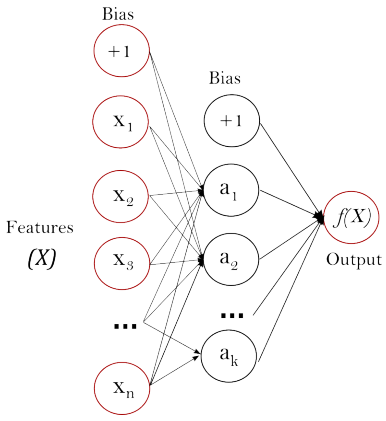

Figure 12: One hidden layer MLP

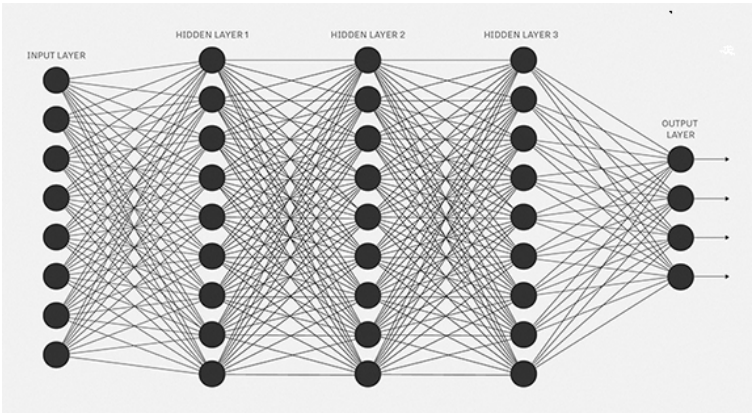

Figure 13: A neural network made of interconnected neurons (dense layer)

Layer (type)	Output Shape	Param #
dense_9 (Dense)	(None, 8)	112
dense_10 (Dense)	(None, 4)	36
dense_11 (Dense)	(None, 1)	5

Total params: 153
Trainable params: 153
Non-trainable params: 0

Figure 14: Neural network architecture summary for Boston data set

Layer (type)	Output Shape	Param #
dense_6 (Dense)	(None, 16)	384
dense_7 (Dense)	(None, 8)	136
dense_8 (Dense)	(None, 1)	9

Total params: 529
Trainable params: 529
Non-trainable params: 0

Figure 15: Neural network architecture summary for Default data set

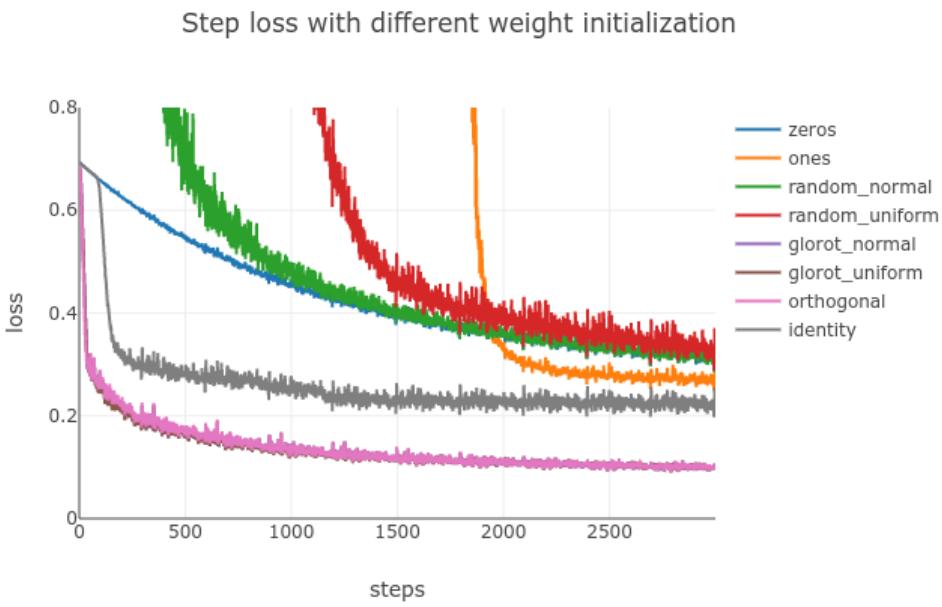

Figure 16: Step loss with different initialisations

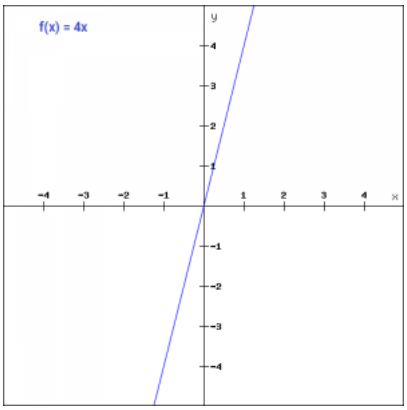

Figure 17: Linear function

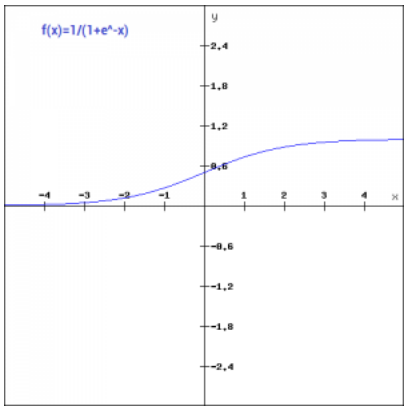

Figure 18: Sigmoid function

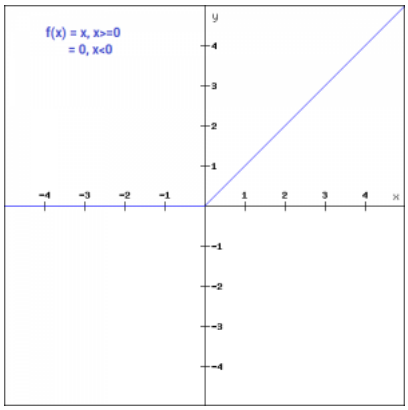

Figure 19: ReLU function

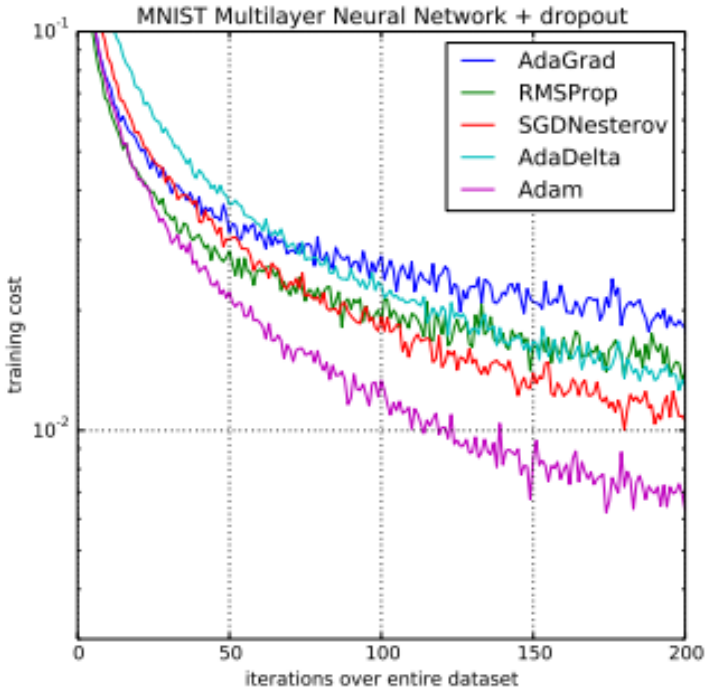

Figure 20: Comparison of Adam to Other Optimization Algorithms Training a Multilayer Perceptron

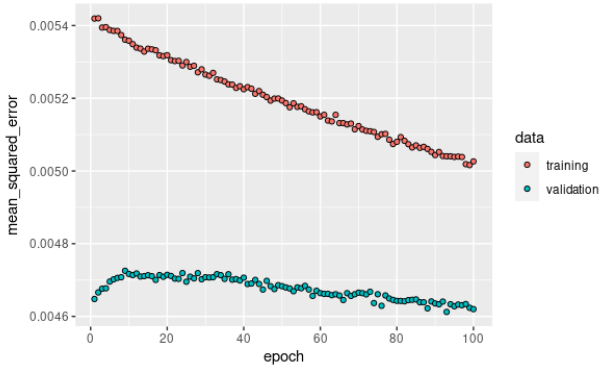

Figure 21: MLP model (Boston)

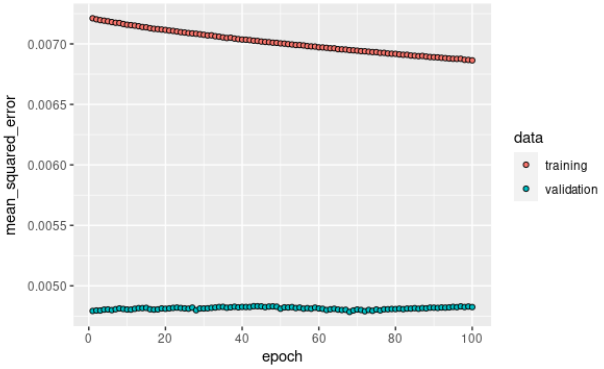

Figure 22: MLP L2 model (Boston)

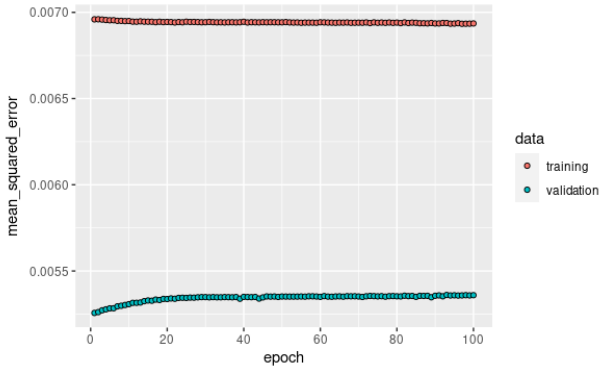

Figure 23: MLP L1 model (Boston)

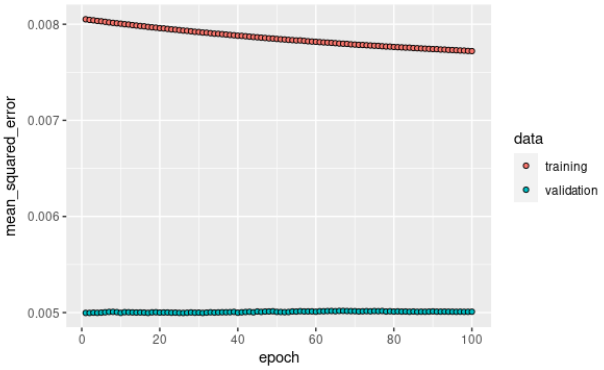

Figure 24: MLP L1_L2 model (Boston)

```
> # MODELS COMPARISION ------------------------------------------------------------
> # Compute the average train/val MSE for all folds of 4 models
>
> paste0("Average normalised MSE of MLP on validation set: $", sprintf("%.2f", mean(mse_histories_mlp)))
[1] "Average normalised MSE of MLP on validation set: $0.01"
> paste0("Average normalised MSE of MLP (L2 regularisation) on validation set: $", sprintf("%.2f", mean(mse_
histories_mlp_l2)))
[1] "Average normalised MSE of MLP (L2 regularisation) on validation set: $0.01"
> paste0("Average normalised MSE of MLP (L1 regularisation) on validation set: $", sprintf("%.2f", mean(mse_
histories_mlp_l1)))
[1] "Average normalised MSE of MLP (L1 regularisation) on validation set: $0.02"
> paste0("Average normalised MSE of MLP (L1 and L2 regularisation) on validation set: $", sprintf("%.2f", me
an(mse_histories_mlp_l1_l2)))
[1] "Average normalised MSE of MLP (L1 and L2 regularisation) on validation set: $0.02"
>
```

Figure 25: Average normalised MSE of MLP models on Boston validation data set

```
> set.seed(1)
> scaled_Y_hat <- predict(mlp, scaled_X_test)
> Y_hat = denormalizeData(scaled_Y_hat, getNormParameters(scaled_Y_test))
> MSE = mean((Y_test - Y_hat)^2)
> R2 = 1 - sum((Y_test - Y_hat)^2)/sum((Y_test - mean(Y_test))^2)
> paste("MSE:", MSE, "R squared:", R2)
[1] "MSE: 67.1301435926387 R squared: 0.388284976627279"
>
```

Figure 26: Final MSE of MLP model on Boston test data set

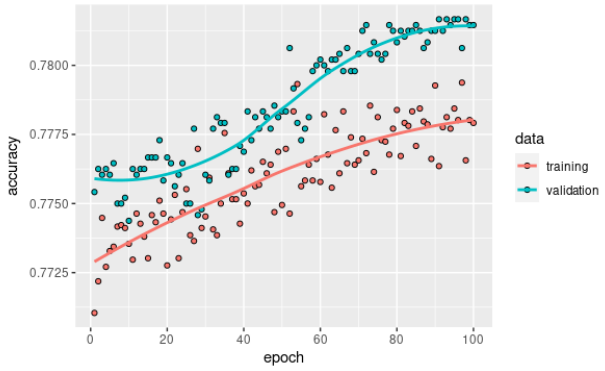

Figure 27: MLP model (Default)

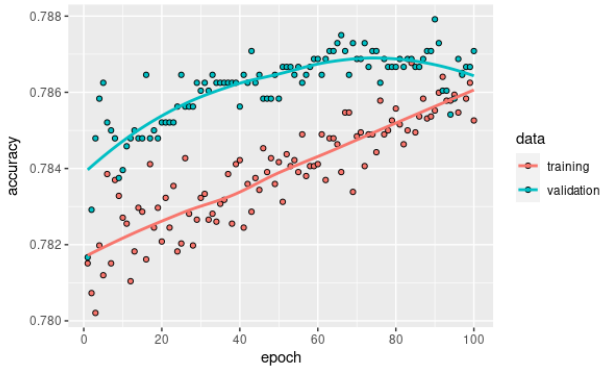

Figure 28: MLP L2 model (Default)

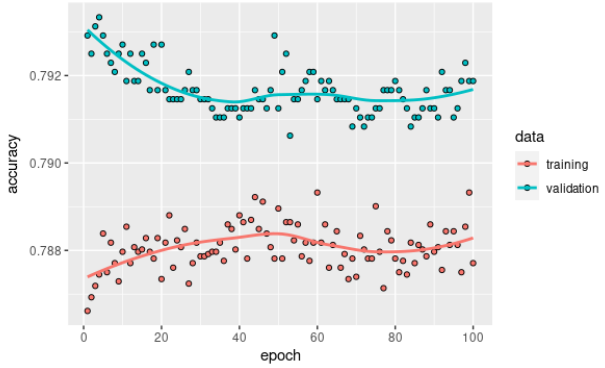

Figure 29: MLP L1 model (Default)

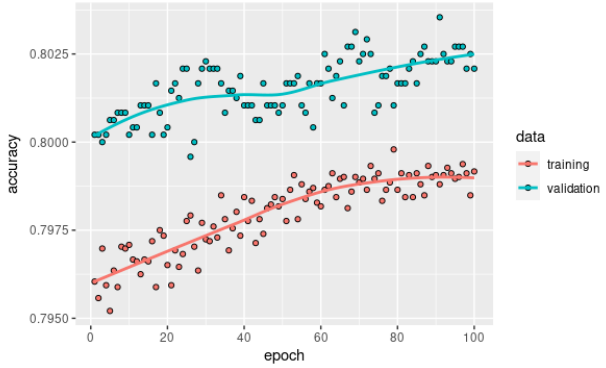

Figure 30: MLP L1_L2 model (Default)

```
> paste0("Average accuracy of MLP on validation set: ", sprintf("%.2f", mean(acc_histories_mlp)))
[1] "Average accuracy of MLP on validation set: 0.75"
> paste0("Average accuracy of MLP (L2 regularisation) on validation set: ", sprintf("%.2f", mean(acc
_histories_mlp_l2)))
[1] "Average accuracy of MLP (L2 regularisation) on validation set: 0.76"
> paste0("Average accuracy of MLP (L1 regularisation) on validation set: ", sprintf("%.2f", mean(acc
_histories_mlp_l1)))
[1] "Average accuracy of MLP (L1 regularisation) on validation set: 0.77"
> paste0("Average accuracy of MLP (L1 and L2 regularisation) on validation set: ", sprintf("%.2f", m
ean(acc_histories_mlp_l1_l2)))
[1] "Average accuracy of MLP (L1 and L2 regularisation) on validation set: 0.78"
>
```

Figure 31: Average accuracy of MLP models on Default validation data set

```
> #------------------------------------------------------------------------
> ## 6. TEST NN MODEL
> #------------------------------------------------------------------------
> set.seed(1)
> pred_mlp <- predict_classes(mlp, scaled_X_test)
> table(Y_test, pred_mlp)
      pred_mlp
Y_test    0    1
     0 3913  723
     1  641  723
> mean(Y_test == pred_mlp)
[1] 0.7726667
>
> set.seed(1)
> pred_mlp_l2 <- predict_classes(mlp_l2, scaled_X_test)
> table(Y_test, pred_mlp_l2)
      pred_mlp_l2
Y_test    0    1
     0 3771  865
     1  625  739
> mean(Y_test == pred_mlp_l2)
[1] 0.7516667
>
```

Figure 32: Final confusion matrix of MLP model and MLP L2 models on Default test data set

```
> set.seed(1)
> pred_mlp_l1 <- predict_classes(mlp_l1, scaled_X_test)
> table(Y_test, pred_mlp_l1)
      pred_mlp_l1
Y_test    0    1
     0 3906  730
     1  633  731
> mean(Y_test == pred_mlp_l1)
[1] 0.7728333
>
> set.seed(1)
> pred_mlp_l1_l2 <- predict_classes(mlp_l1_l2, scaled_X_test)
> table(Y_test, pred_mlp_l1_l2)
      pred_mlp_l1_l2
Y_test    0    1
     0 3941  695
     1  656  708
> mean(Y_test == pred_mlp_l1_l2)
[1] 0.7748333
```

Figure 33: Final confusion matrix of MLP L1 model and MLP L1_L2 models on Default test data set

```
> summary(df)
    default          Baseline          L2               L1           L1.and.L2
 Min.   :0.0000   Min.   :0.000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
 Median :0.0000   Median :0.000   Median :0.0000   Median :0.0000   Median :0.0000
 Mean   :0.2273   Mean   :0.241   Mean   :0.2673   Mean   :0.2435   Mean   :0.2338
 3rd Qu.:0.0000   3rd Qu.:0.000   3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.0000
 Max.   :1.0000   Max.   :1.000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
> save.image(" /default_mlp_RData")
```

Figure 34: Summary of MLP models' result and actual result on Default test data set

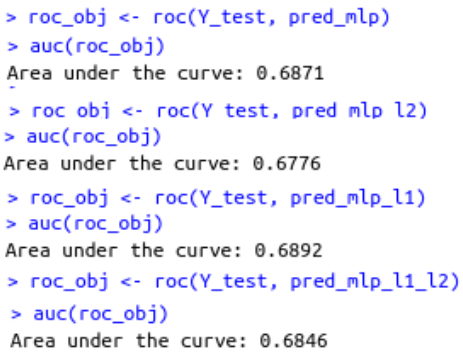

Figure 35: Final AUC of MLP models on Default test data set