Credit Card Default Data

SEIS 763-Machine Learning and Deep Learning

Alebachew Banteaymolu | Elizabeth Tesfa.G | Mariyam George

Prerna Sawanat | Sai Ratnam Peri

Introduction

This project is aimed at the case of customers default payments in Taiwan and was used to compare the predictive accuracy of probability of default using Machine Learning techniques.

Tools and Data Preprocessing

		Purpose
MATLAB		Data Pre-Processing and Manipulation
Tableau		Descriptive Analysis
Excel		Initial Data Visualization
Pycharm CE	Neural Networks	
H2o package	Neural Networks	
Microsoft Word		Word Processing

Data source

We used the Credit card default data found on https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset. This dataset 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.year with as many as 25 attributes.

The attributes include:

- 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)
- PAY_AMT3: Amount of previous payment in July, 2005 (NT dollar)
- 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)

Data Manipulation and preprocessing:

The data that was selected is in native XLS (Microsoft Excel) file format. We preliminarily used R and XLS to run some of the descriptive analysis such as data size, attributes count. Our intend was to have no missing data. There were a couple challenges with the existing raw data, first challenge was there was a lot of confusion with the column names as a part of preprocessing, to avoid confusion we changed some column names. Some of them include:

PAY_0 to SEPT REPAY STS

PAY_2 to AUG REPAY STS

PAY_3 to JULY REPAY STS

PAY_4 to JUNE REPAY STS

PAY_5 to MAY REPAY STS

PAY_6 to APRIL REPAY STS

BILL_AMT1 to SEPT STMT

BILL_AMT2 to AUG STMT

BILL_AMT3 to JULY STMT

BILL_AMT4 to JUNE STMT

BILL_AMT5 to MAY STMT

BILL_AMT6 to APRIL STMT

PAY_AMT1 to SEPT AMT.

PAY_AMT2 to AUG AMT.

PAY_AMT3 to JULY AMT.

PAY_AMT4 to JUNE AMT.

PAY_AMT5 TO MAY AMT.

PAY_AMT6 to APRIL AMT.

The second problem was negative values in the data.Though it was mentioned in metadata, we thought of converting into positive values so that understanding the results would be much easier with positive data than with the negative one.Thus we converted zero to two, -1 to 1 and -2 to 0 using MATLAB.

As some of the categorical variables contain numerical values that range from -2 to 8, the values had to be changed into positive value before changing to dummy variable.This is done using MATLAB.

```
Data.SEPTREPAYSTS = Data.SEPTREPAYSTS + 3;
Data.APRREPAYSTS = Data.APRREPAYSTS + 3;
Data.AUGREPAYSTS = Data.AUGREPAYSTS + 3;
Data.JULYREPAYSTS = Data.JULYREPAYSTS + 3;
Data.JUNEREPAYSTS = Data.JUNEREPAYSTS + 3;
Data.MAYREPAYSTS = Data.MAYREPAYSTS + 3;
```

Other categorical variables have values that are not listed in the data description, hence these had to be changed to ‘other’.

```
tmp = Data.EDUCATION; % Education has variables that are not in the data description and hence these values are moved to 'Other' (4)

tmp((tmp == 0 | tmp == 5 | tmp == 6)) = 4;

Data.EDUCATION = tmp;

 marital = Data.MARRIAGE;  % Marriage has value ‘0’ and this is changed to other (3)

marital(marital == 0) = 3;

Data.MARRIAGE = marital;
```

Once the pre-processing is done, categorical predictors are changed into Dummy variable and numerical predictors are standardized as follows:

```
sex = dummyvar(Data.SEX);

education = dummyvar(Data.EDUCATION);

marriage = dummyvar(Data.MARRIAGE);

septRepSts = dummyvar(Data.SEPTREPAYSTS);

augRepSts = dummyvar(Data.AUGREPAYSTS);

julRepSts = dummyvar(Data.JULYREPAYSTS);

junRepSts = dummyvar(Data.JUNEREPAYSTS);

mayRepSts = dummyvar(Data.MAYREPAYSTS);

aprRepSts = dummyvar(Data.APRREPAYSTS);


Xnum = [Data.LIMIT_BAL, Data.AGE, Data.SEPTSTMT, Data.AUGSTMT,
Data.JULYSTMT, Data.JUNESTMT, Data.MAYSTMT, Data.APRSTMT, Data.SEPTAMT,
Data.AUGAMT, Data.JULYAMT, Data.JUNEAMT, Data.MAYAMT, Data.APRAMT];

Xstnd = zscore(Xnum);
```

Finally, the standardized and normalized data is then put together to form the predictors and response variable as shown below:

```
X = [Xstnd(:,1), sex(:,2), education(:,2:4), marriage(:,2:3), Xstnd(:,2), septRepSts(:,2:11),
augRepSts(:,2:11), julRepSts(:,2:11), junRepSts(:,2:11), mayRepSts(:,2:11),
aprRepSts(:,2:11), Xstnd(:,3:14)];

Y = [Data.default];
predictors = [X, Y];
```

Dividing data into training, testing and validation

Before applying any of the classification algorithms in our dataset, the prepared dataset is changed into testing, training datasets as shown below:

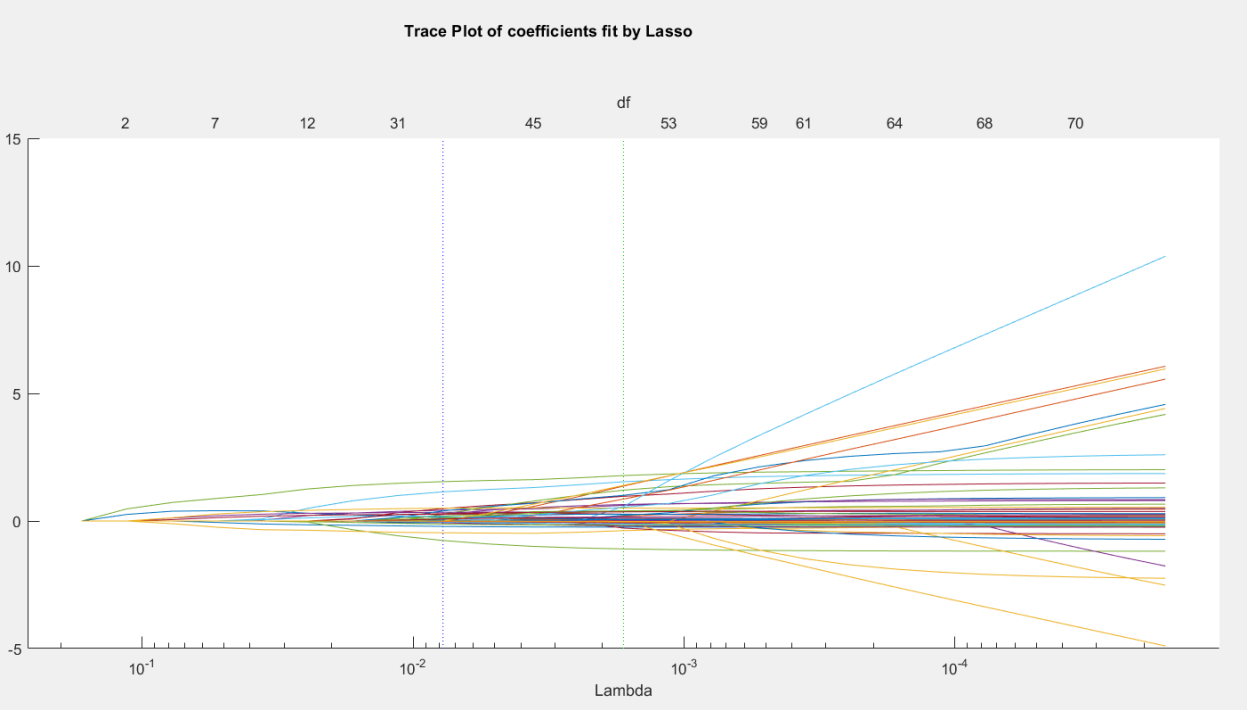
```
testData = table2array(predictors)';
% Predictors is 30000x81 data with the last column being the response variable
[training, testing] = dividerand(testData,0.8,0.2);
% Training (80%), Testing (20%)
training = training';
testing = testing';
dlmwrite('training.csv',training);
dlmwrite('testing.csv',testing);
```

Logistic Regression and Regularization(Lasso)

Code:

```
%Reading xlx file
trainData = csvread("U:\Project\trainBal.csv");
X = trainData(:,1:80);
Y = trainData(:,81);
LoMdl = fitglm(X, Y, 'distr', 'binomial', 'link', 'logit')  %Logistic

[b fitinfo] = lassoglm(X, Y,  'binomial','NumLambda',25,'CV',10, 'Alpha', 1); %Lasso with all predictors (80)

lassoPlot(b,fitinfo,'PlotType','Lambda','XScale','log');
lassoPlot(b, fitinfo, 'PlotType', 'CV');

trainData(:,[17,18,20,23,24,27,28,30,31,34,35,38,40,41,45,46,48,51,56,57,58,61,64,65,67,68,69,72,73,74]) =[];

testData = csvread("U:\Project\testBal.csv");
testData(:,[17,18,20,23,24,27,28,30,31,34,35,38,40,41,45,46,48,51,56,57,58,61,64,65,67,68,69,72,73,74]) =[];


[b1 fitinfo1] = lassoglm(trainData(:,1:50), trainData(:,51), 'binomial','NumLambda',25,'CV',10, 'Alpha', 1); %Lasso after regularization(removing 30 predictors)

lassoPlot(b1,fitinfo1,'PlotType','Lambda','XScale','log');
```

Figure: Lasso with Lambda before regularization



Dummy Variables as represented

Numerical labels were given for each categorical field in the data and we didn't change the labels when we convert them in to dummy variables.

- Gender (1 = male; 2 = female)
  The reference is 1 which is Male There are 18,888 of Male and 11,112 Female
- Education (1 = graduate school; 2 = university; 3 = high school; 4 = others)
  The reference is 1 which is graduate school 1 = university; 2 = high school; 3 = others).
- Marital status (1 = married; 2 = single; 3 = others)
  The reference is 1 which is married
- **payment status for six months from months April September.**
- The data tracked the past monthly payment records (from April to September,) as follows: X6 = the repayment status in September; X7 = the repayment status in August, 2005; . . .; X11

  In this category we found there were some unlabeled data and we assumed that the -2 label means paid early as a result we have 11 categories

  In the original data the measurement scale for the repayment status is: -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. We added 3 to each label (refer the code) and hence, the reference for these categories is **Early Paid and** the rest categories are pay duly, payment delay for one month, payment delay for two months and so forth.

Using Lasso:
30 predictors were found as insignificant to train …

To mention few of the indicator which contributes less to train for default are:
Column (17,18) ---Payments delayed by 7 and 8 months in the month of September
Column (35,38) Payments delayed by 6 and 8 months in the month of July

Column (64,65,67,68) payments delayed by 5,6,8 and 9-month delay in April
We remove these 30 predictors and rerun Lasso it the performance didn't change which indicates that they are insignificant to the overall performance.

Figure: Lasso with Lambda after regularization

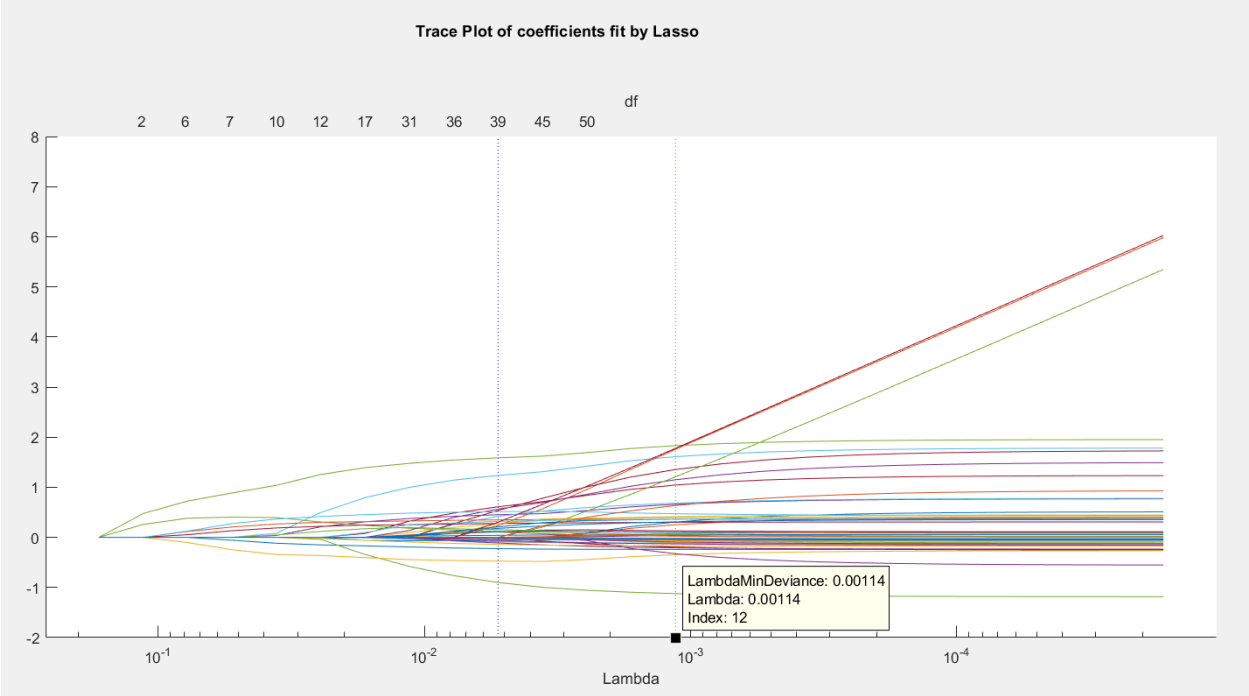
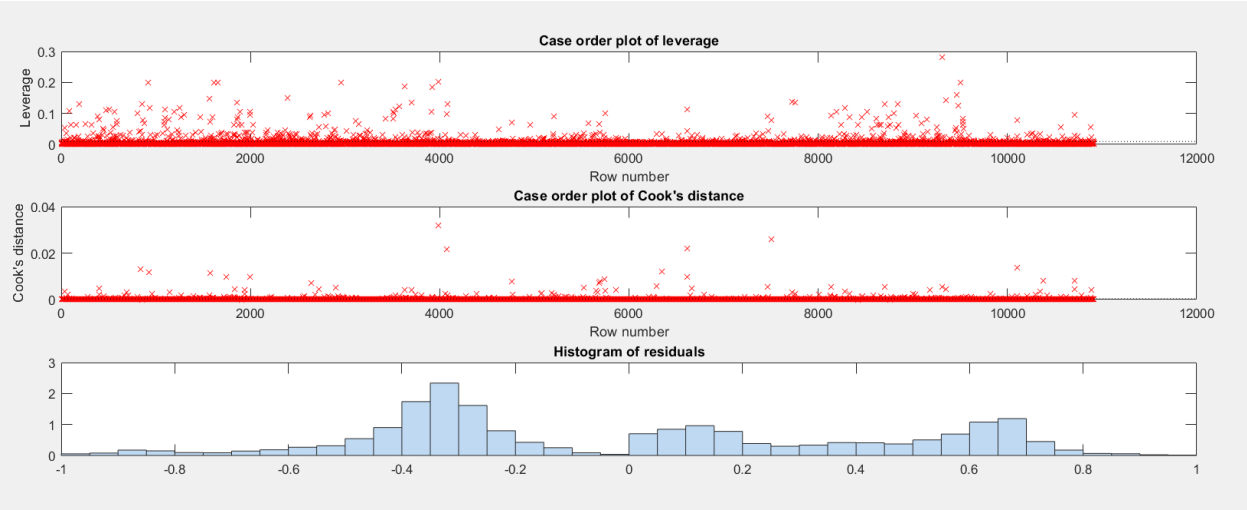


Figure: Cooks distance, leverage after reducing predictors to 50



KNN(Nearest Neighbours)

Taking the above test and training data,performance of KNN is as follows:

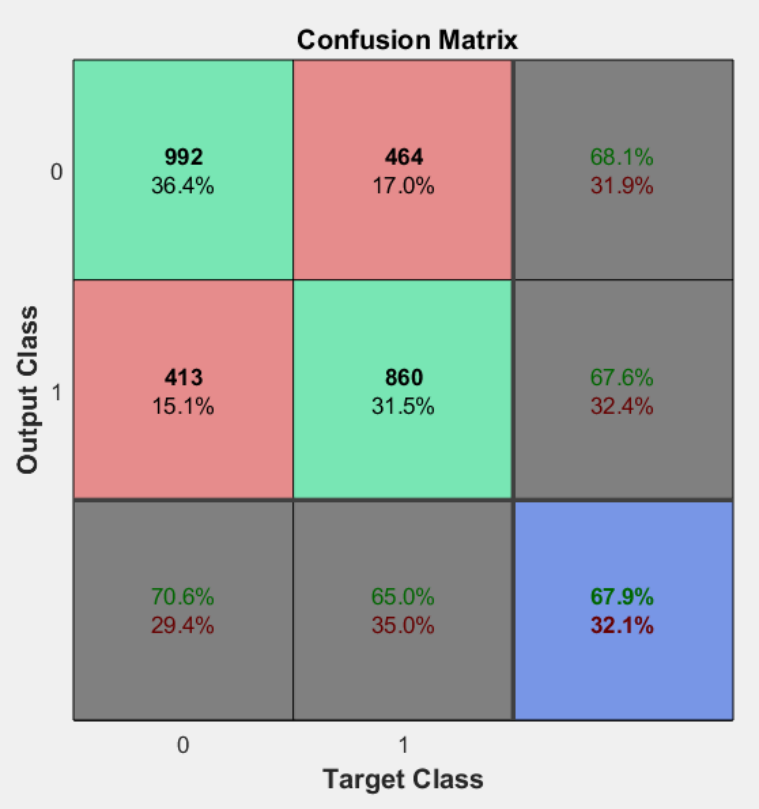
```
%Reading xlx file
trainData = csvread("U:\Project\trainBal.csv");
X = trainData(:,1:50);
Y = trainData(:,51);
Ones = find(Y); % This returns the index for all '1's
Zeros = find(~Y); % This returns the index for all '0's
size(Ones)
size(Zeros)

KnnMdl = fitcknn(X,Y,'NumNeighbors',7,'Standardize',1);
KnnMdl.ClassNames
KnnMdl.Prior

testData = csvread("U:\Project\testBal.csv");
testData(:,[17,18,20,23,24,27,28,30,31,34,35,38,40,41,45,46,48,51,56,57,58,61,64
,65,67,68,69,72,73,74]) =[];

XTest = testData(:,1:50);
```

%Plotting confusion Matrix for Test data

```
[n,p] = size(XTest)
isLabels = unique(YTest);
nLabels = numel(isLabels)
[~,grpOOF] = ismember(label,isLabels);
oofLabelMat = zeros(nLabels,n);
idxLinear = sub2ind([nLabels n],grpOOF,(1:n)');
oofLabelMat(idxLinear) = 1; % Flags the row corresponding to the class
[~,grpY] = ismember(YTest,isLabels);
YMat = zeros(nLabels,n);
idxLinearY = sub2ind([nLabels n],grpY,(1:n)');
YMat(idxLinearY) = 1;

figure;
plotconfusion(YMat,oofLabelMat);
h = gca;
h.XTickLabel = [num2cell(isLabels); {''}];
h.YTickLabel = [num2cell(isLabels); {''}];
```

```
%ROC class 0
[xpos, ypos, ~, AUC0] = perfcurve(YTest, score(:, 1), 0); % For the first class.
figure, plot(xpos, ypos); % Plot graph
xlim([-0.05 1.05]), ylim([-0.05 1.05]);
xlabel('\bfFPrate'), ylabel('\bfTPrate'); % insert X and Y labels
title('\bf ROC for class 0 by KNN'); % insert main title
legend('AUC 0');
%ROC class 1
[xpos, ypos, ~, AUC1] = perfcurve(YTest, score(:, 2), 1); % For the second class.
figure, plot(xpos, ypos);
xlim([-0.05 1.05]), ylim([-0.05 1.05]);
xlabel('\bfFPrate'), ylabel('\bfTPrate');
title('\bf ROC for class 1 by KNN');
legend('AUC 1');
```

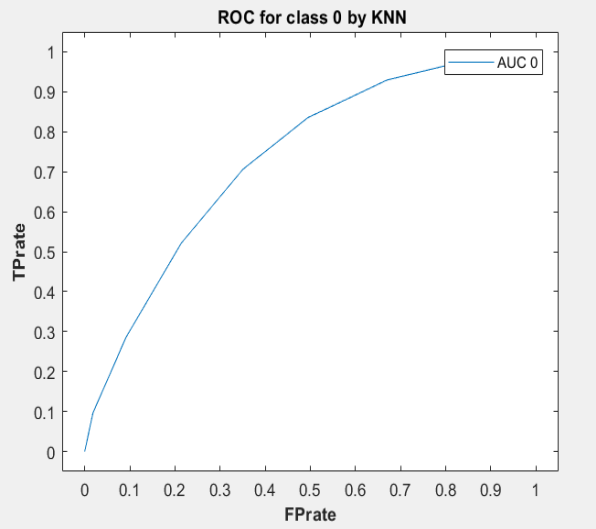
Confusion Matrix:

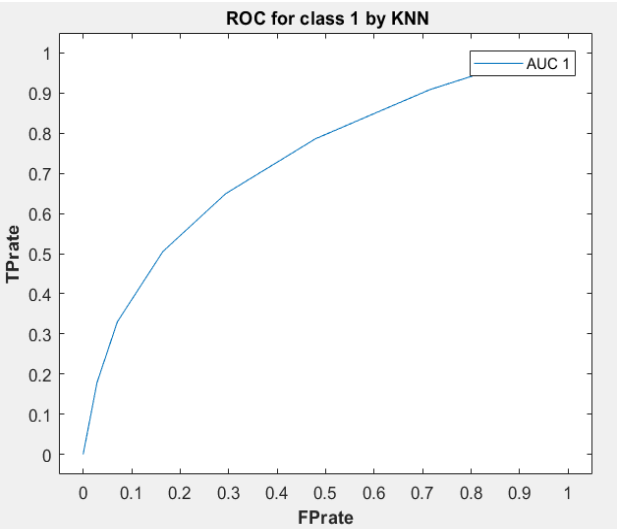


Summary KNN:

Measurements	Classes	
	0	1
Accuracy	0.6786	0.6786
Precision	0.6813	0.6756
Recall	0.7060	0.6495
F1_Score	0.6935	0.6623

ROCS KNN:


ROC for class 0 by KNN
AUC 0
TPrate
FPrate
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


ROC for class 1 by KNN
AUC 1
TPrate
FPrate
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Neural network(NN):

```
#Importing h2o package to perfrom deep learning
import h2o
from h2o.estimators.deeplearning import H2ODeepLearningEstimator
import pandas as pd

h2o.init(nthreads = -1, max_mem_size = 8)

# Assigning predictors to x dataframe
#Reading training CSV file as a dataframe using Pandas
data_training = pd.read_csv('/Users/bonythomas/RtrainData.csv')
#Reading testing CSV file as a dataframe using Pandas
data_testing = pd.read_csv('/Users/bonythomas/RtestData.csv')
#Converting the dataframe to H2o dataframe
data_training=h2o.H2OFrame(data_training.values.tolist())
data_testing=h2o.H2OFrame(data_testing.values.tolist())
#Factorizing the target value for classification
data_training['C51'] = data_training['C51'].asfactor()
data_training['C51'].levels()

#Assigning the target column header to y
y = 'C51'
#Assigning the predictors header to x
x = list(data_training.columns)
#remove the response from x
x.remove(y)

# Initialize and train the DL estimator:
#One Epoch With hidden  layer
dl_1 =
H2ODeepLearningEstimator(hidden=[200,200],epochs=1, activation="Tanh",loss = "crossentropy"
)
dl_1.train(x=x, y=y,training_frame=data_training)

#250 epochs with hidden layers
dl_2 = dl_250 =
H2ODeepLearningEstimator(hidden=[11,13,17,19],checkpoint=dl_1, epochs=250,loss = "crossentro
py")
dl_2.train(x=x, y=y, training_frame=data_training)

#Perfromance of the Nerual network
dl_per2 = dl_2.model_performance()
#Print the perfromance of the model with 250 epochs
print("Pref 2:",dl_per2)
```

Using the same testing and training data we performed NN using h2o with python.

Metric	Thresold	Value	Idx
Max f1	0.00923739	0.708969	369
Max f2	0.00149209	0.82575	396
Max accuracy	0.0201603	0.711781	336
Max precision	0.926457	1	0
Max recall	0.000281579	1	399
Max min_per_class_accuracy	0.0291177	0.431808	349
Max_mean_per_class_accuracy	0.0201603	0.709353	336

Output: performance of model 1 with one epoch(accuracy 71%)

0	1	Error	Rate
2576	2558	0.4982	(2558.0/5134.0)
781	4067	0.1611	(781.0/4848.0)
3357	6625	0.3345	(3339.0/9982.0)

Performance of model 2 with 250 epochs and with different hidden layers(accuracy 73%)

0	1	Error	Rate
3147	1988	0.3871	1988/5135
919	3940	0.1891	919/4859
4066	5928	0.2909	2907/9994

Metric	Thresold	Value	Idx
Max f1	0.965847	0.730509	109
Max f2	0.898677	0.836906	250
Max accuracy	0.981257	0.731639	66
Max precision	0.999962	0.948718	0
Max recall	0.737375	1	376
Max min_per_class_accuracy	0.9756	0.7234	82
Max_mean_per_class_accuracy	0.981257	0.729808	66

Summary of model 1:
Support Vector Machine(SVM):

Using these datasets, we run SVM classification as showing below.

```
Xtrain = table2array(training(:,1:80));
Ytrain = table2array(training(:,81));
Xtest = table2array(testing(:,1:80));
Ytest = table2array(testing(:,81));
svm_mdl = fitcsvm(Xtrain, Ytrain, 'KernelFunction', 'rbf', 'Crossval', 'on');
[label, score] = predict(svm_mdl.Trained{10,1}, Xtest);
```

Next, calculate accuracy, precision and recall for each class.

cfm =

confusionmat(Ytest, label)
cfm =

4532	120
1118	230

```
accuracy = sum(diag(cfm))/sum(cfm(:));
prcsn0 = cfm(1,1)/sum(cfm(:,1));
recall0 = cfm(1,1)/sum(cfm(1,:));
F1_0 = (2*prcsn0*recall0)/(recall0 + prcsn0);
prcsn1 = cfm(2,2)/sum(cfm(:,2));
recall1 = cfm(2,2)/sum(cfm(2,:)); % calculate recall for class 1
F1_1 = (2*prcsn1*recall1)/(recall1 + prcsn1);
```

summary:

Measurements	Classes	
	0	1
Accuracy	0.7937	0.7937
Precision	0.8021	0.6571
Recall	0.9742	0.1706
F1_Score	0.8798	0.2709

Though SVM performed very well for class '0', we can see that the measurements for class '1' recall and F1 score are very low. This shows that there is 00an imbalance (skewed) data problem in the response variable (also shown in the confusion matrix) and this could have a huge effect on classification algorithms.

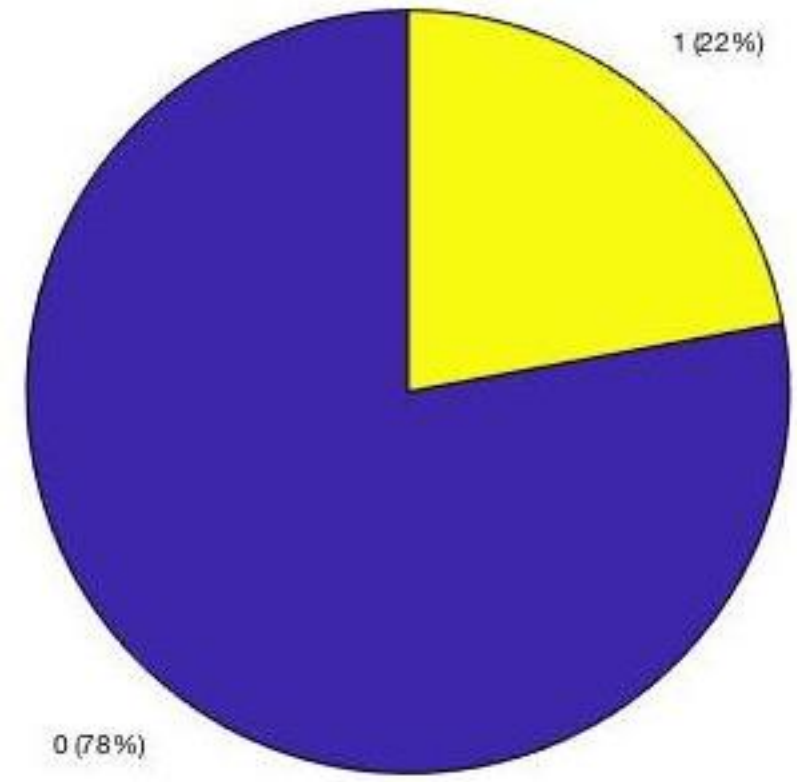


Figure 1: As shown above, the binary response variables are hugely unbalanced and this would lead to a majority/minority issue.

Though there could be a potential information loss, one way to resolve the imbalance data issue would be to apply under sampling on '0's and oversampling on '1' response variables.

```
testData = table2array(predictors); % Original data

Y = testData(:,81); % Response variables

Ones = find(Y); % This returns the index for all '1's

Zeros = find(~Y); % This returns the index for all '0's

size(Ones)
        6636  1
size(Zeros)
        23364  1


A = testData(Ones,:); % Get all rows with '1'  response variable

B = testData(Zeros,:); % Get all rows with '0' response variable

[newB, ~, ~] = dividerand(B',0.3,0.5,0.2);
newB = newB';
size(newB) % This is closer to the size of '1's
        7009  81
testNew =[A;newB]; % Combine the balanced data.
size(testNew)
        13645  81
```

Next, we divide this balanced data into training, testing

```
[training, ~, testing] = dividerand(testNew', 0.8, 0.0, 0.2); % Training (80%) and Testing (20%)

training = training';

testing = testing';
```

Re-run the SVM classification algorithm.

```
svm_mdl = fitcsvm(Xtrain, Ytrain, 'KernelFunction', 'rbf', 'Crossval', 'on');
[label, score] = predict(svm_mdl.Trained{10,1}, Xtest);
cfm = confusionmat(Ytest, label);
accuracy = sum(diag(cfm))/sum(cfm(:));
prcsn0 = cfm(1,1)/sum(cfm(:,1));
recall0 = cfm(1,1)/sum(cfm(1,:));
F1_0 = (2*prcsn0*recall0)/(recall0 + prcsn0);
prcsn1 = cfm(2,2)/sum(cfm(:,2));
recall1 = cfm(2,2)/sum(cfm(2,:)); % calculate recall for class 1
F1_1 = (2*prcsn1*recall1)/(recall1 + prcsn1);

[xpos, ypos, ~, AUC0] = perfcurve(Ytest, score(:, 1), 0); % For the first class.
figure, plot(xpos, ypos); % Plot graph
xlim([-0.05 1.05]), ylim([-0.05 1.05]);
xlabel('\bfFPrate'), ylabel('\bfTPrate'); % insert X and Y labels
title('\bf ROC for class 0 by SVM'); % insert main title
legend('AUC = 0.708');

[xpos, ypos, ~, AUC1] = perfcurve(Ytest, score(:, 2), 1); % For the second class.
figure, plot(xpos, ypos);
xlim([-0.05 1.05]), ylim([-0.05 1.05]);
xlabel('\bfFPrate'), ylabel('\bfTPrate');
title('\bf ROC for class 1 by SVM');
legend('AUC = 0.708');
```

Confusion matrix:

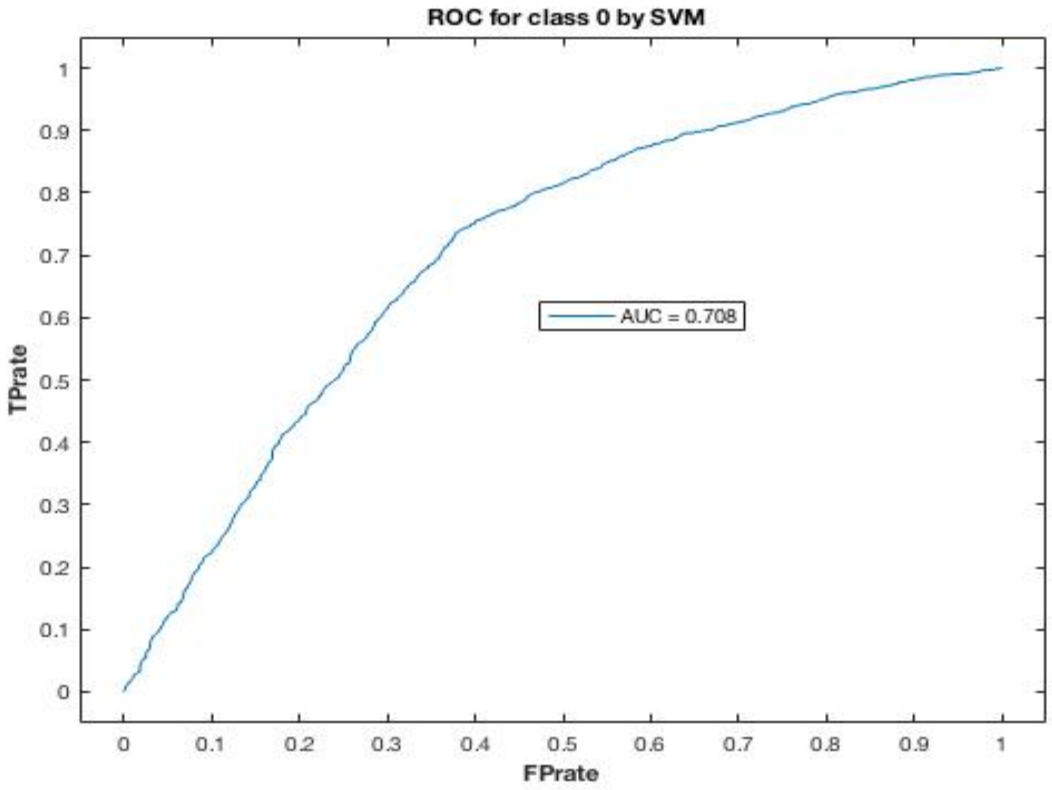
1333 794

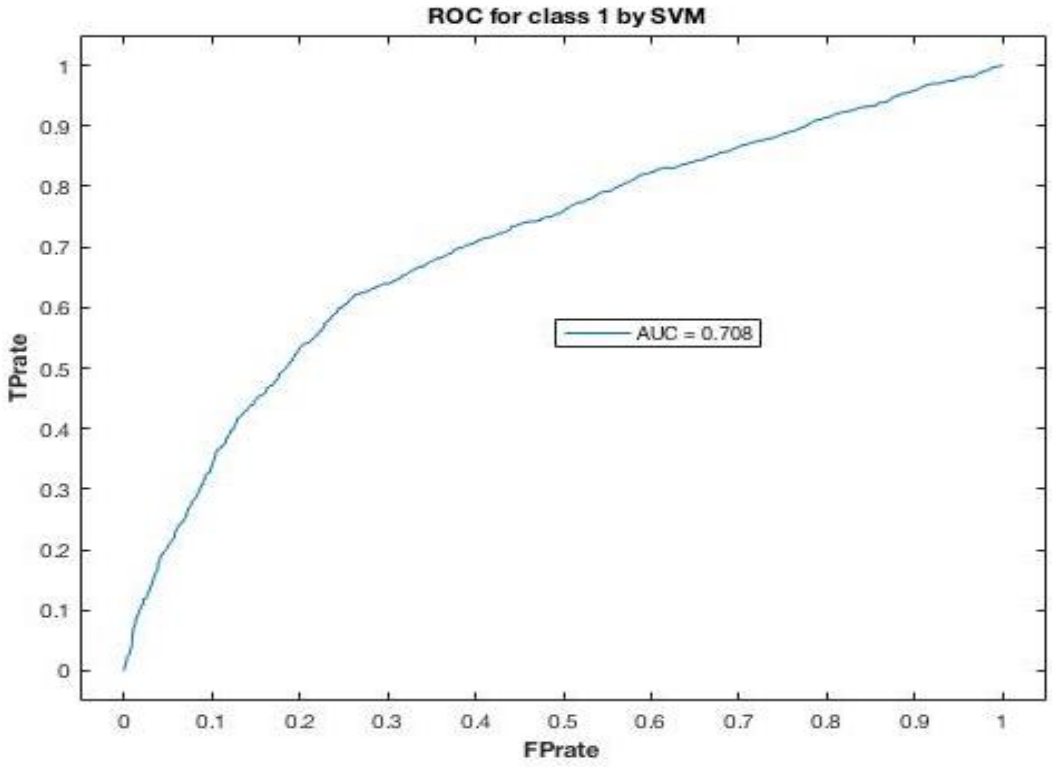
608 1359

Summary:

Measurements	Classes	
	0	1
Accuracy	0.6575	0.6575
Precision	0.6868	0.6312
Recall	0.6267	0.6909
F1_Score	0.6554	0.6597

As shown in the table above, there is a huge recall and F1 score improvement for class '1' though at the expense information lose in class '0'.





After applying lasso and reducing the number of predictors from 80 to 50.

Confusion matrix

1110 295
531 793

Summary:

Measurements	Classes	
	0	1
Accuracy	0.6973	0.6973
Precision	0.6767	0.7289
Recall	0.7900	0.6012
F1_Score	0.7288	0.6575

Confusion matrix:

Measurements	Class 0 : No Default				Class 1 : Yes Default			
	Logistic Regression	KNN	SVM	NN	Logistic Regression	KNN	SVM	NN
Accuracy	70.03	67.86	69.73	70.56	70.03	67.86	69.73	70.56
Precision	82.92	75.56	67.67	82.51	56.34	61.19	72.89	56.23
Recall	66.84	67.42	79.00	66.73	75.66	66.88	60.12	66.32
F Score	74.02	71.77	72.88	70.36	64.59	63.38	65.75	62.71
ROC - AUC	76.97	74.46	72.72	75.49	76.97	74.46	72.72	72.67

Here is the combined ROC curves for all the models we used.