NUS
National University
of Singapore

GROUP 15

BIG DATA ANALYTICS USING ANN

INTRODUCTION

- WHAT IS OUR PROJECT ABOUT ?

  - A Bank, where the customers are closing down their accounts due to certain number of factors and we don’t know which factors are most influencing ones.

  - In this project we tried to investigate that which factors are responsible for the bank losing customers using ANN and experimented using different techniques so that after comparison we can get the best output possible.

  - The different techniques used are :

    - Naive Bayes
    - Stochastic gradient descend algorithm ADAM
    - Back Propagation
    - Resilient Back Propagation

DATA

- DATASET WE WORKED ON :
  - We have a Database consisting of 10000 entries.
  - The different fields in the dataset are :

CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited

- **Customer ID:**

A unique identification number, generated at time of account opening

- **Exited**:

Its our response variable and in the data set it tells us whether the

Customer has exited or not.

- **Estimated Salary :**

Its the estimated salary of the customer.

- **Tenure :**

For how long the customer has been linked with the bank.

- **Has Cr Card:**

Whether the customer has credit card or no.

- **Num Of Products:**

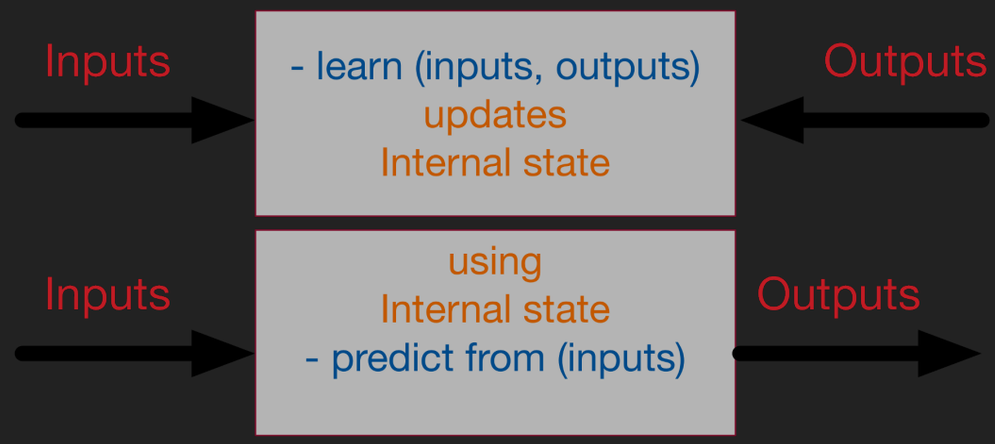
How many services did he avail from the bank.

4.

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
1	15634602	Hargrave	619	France	Female	42	2	0	1	1	1	101348.88	1
2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
3	15619304	Onio	502	France	Female	42	8	159660.8	3	1	0	113931.57	1
4	15701354	Boni	699	France	Female	39	1	0	2	0	0	93826.63	0
5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.1	0
6	15574012	Chu	645	Spain	Male	44	8	113755.78	2	1	0	149756.71	1
7	15592531	Bartlett	822	France	Male	50	7	0	2	1	1	10062.8	0
8	15656148	Obinna	376	Germany	Female	29	4	115046.74	4	1	0	119346.88	1
9	15792365	He	501	France	Male	44	4	142051.07	2	0	1	74940.5	0
10	15592389	H?	684	France	Male	27	2	134603.88	1	1	1	71725.73	0
11	15767821	Bearce	528	France	Male	31	6	102016.72	2	0	0	80181.12	0
12	15737173	Andrews	497	Spain	Male	24	3	0	2	1	0	76390.01	0
13	15632264	Kay	476	France	Female	34	10	0	2	1	0	26260.98	0
14	15691483	Chin	549	France	Female	25	5	0	2	0	0	190857.79	0
15	15600882	Scott	635	Spain	Female	35	7	0	2	1	1	65951.65	0
16	15643966	Goforth	616	Germany	Male	45	3	143129.41	2	0	1	64327.26	0
17	15737452	Romeo	653	Germany	Male	58	1	132602.88	1	1	0	5097.67	1
18	15788218	Henderson	549	Spain	Female	24	9	0	2	1	1	14406.41	0
19	15661507	Muldrow	587	Spain	Male	45	6	0	1	0	0	158684.81	0
20	15568982	Hao	726	France	Female	24	6	0	2	1	1	54724.03	0
21	15577657	McDonald	732	France	Male	41	8	0	2	1	1	170886.17	0
22	15597945	Dellucci	636	Spain	Female	32	8	0	2	1	0	138555.46	0
23	15699309	Gerasimov	510	Spain	Female	38	4	0	1	1	0	118913.53	1
24	15725737	Mosman	669	France	Male	46	3	0	2	0	1	8487.75	0
25	15625047	Yen	846	France	Female	38	5	0	1	1	1	187616.16	0
26	15738191	Maclean	577	France	Male	25	3	0	2	0	1	124508.29	0
27	15736816	Young	756	Germany	Male	36	2	136815.64	1	1	1	170041.95	0
28	15700772	Nebechi	571	France	Male	44	9	0	2	0	0	38433.35	0
29	15728693	McWilliams	574	Germany	Female	43	3	141349.43	1	1	1	100187.43	0
30	15656300	Lucciano	411	France	Male	29	0	59697.17	2	1	1	53483.21	0
31	15589475	Azikiwe	591	Spain	Female	39	3	0	3	1	0	140469.38	1
32	15706552	Odinakachukwu	533	France	Male	36	7	85311.7	1	0	1	156731.91	0

ANALYSIS PROCESS

- OUR APPROACH :
  - A supervised neural network, at the highest and simplest abstract representation, can be presented as a black box with 2 methods learn and predict as following:



  - ***Classification*** is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known.
  - The job of the classification algorithm is to discover how that set of attributes reaches its conclusion.
  - The Predictor columns are use to determine the value of the “predictor attribute”/ resultant variable.

DATA PREPARATION

- *LABEL ENCODER* : Label encoding is simply converting each value in a column to a number.

- *ONE HOT ENCODER* : In this the integer encoded variable is removed and a new binary variable is added for each unique integer value.

- *STANDARD SCALING* : Gaussian with zero mean and unit variance.

- *DATA SPLITTING* : Is done into 80-20. 80% as the training set and 20% is used for validation and testing.

2. BACK PROPAGATION :

- Backpropagation is a method used in artificial neural networks to calculate a gradient that is needed in the calculation of the weights to be used in the network. It is commonly used to train deep neural networks.

- Backpropagation is commonly used by the gradient descent optimization algorithm to adjust the weight of neurons by calculating the gradient of the loss function. This technique is also sometimes called backward propagation of errors.

RStudio

File Edit Code View Plots Session Build Debug

Go to file/function

Source

Console Terminal

```
> OutPutVsPred = cbind(Bank1$Exited,nn1)
> OutPutVsPred
    [,1] [,2]
1      1    0
2      0    0
3      1    0
4      0    0
5      0    1
6      1    0
7      0    0
8      1    0
9      0    1
10     0    1
11     0    0
12     0    0
13     0    0
14     0    0
15     0    0
16     0    1
17     1    1
18     0    0
19     0    0
20     0    0
21     0    0
22     0    0
23     1    0
24     0    0
25     0    0
26     0    0
27     0    0
28     0    0
29     0    0
30     0    0
31     1    0
32     0    0
33     0    0
34     0    0
35     0    0
36     1    1
37     0    0
38     0    0
39     0    0
40     0    0
41     0    0
42     1    0
43     0    0
44     1    0
45     0    0
46     0    0
47     1    0
48     1    0
49     0    0
50     0    0
51     0    0
52     0    1
53     0    0
54     1    0
```

Type here to search

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

Untitled1* z infert Untitled2* College Untitled3* iris cars airquality Untitled4* Bank1 Untitled

Source on Save Run

```
1  ######################################## Traditional Backpropagation Algorithm ##############################
2
3  nn.bp  = neuralnet(
4                 formula = Exited~CreditScore+Age+Tenure+Balance+NumOfProducts+HasCrCard+IsActiveMember+EstimatedSalary,
5                 data = Bank1 , hidden = 3 , learningrate = 0.001  , algorithm = "backprop" , err.fct = "ce" ,
6                 linear.output = FALSE)
7
8  nn.bp
9
```

9:1 Traditional Backpropagation Algorithm

Console Terminal

```
$result.matrix
                                    1
error                   498.191865243310
reached.threshold         0.009950547677
steps                   863.000000000000
Intercept.to.1layhid1    -0.218490687530
CreditScore.to.1layhid1  -0.001652362539
Age.to.1layhid1           0.433533209292
Tenure.to.1layhid1       -0.108048906932
Balance.to.1layhid1      -1.529876374509
NumOfProducts.to.1layhid1 1.879873572512
HasCrCard.to.1layhid1     0.353084951086
IsActiveMember.to.1layhid1 -0.207638677550
EstimatedSalary.to.1layhid1 -0.245711266406
Intercept.to.1layhid2    -0.432067699336
CreditScore.to.1layhid2  -0.277850755889
Age.to.1layhid2           0.018396004292
Tenure.to.1layhid2        0.178517219466
Balance.to.1layhid2      -0.836417987310
NumOfProducts.to.1layhid2 -0.544477052179
HasCrCard.to.1layhid2     0.174802598386
IsActiveMember.to.1layhid2 0.666555160973
EstimatedSalary.to.1layhid2 1.194301396307
Intercept.to.1layhid3     1.362881962541
CreditScore.to.1layhid3   0.544672708130
Age.to.1layhid3          -1.451399748610
Tenure.to.1layhid3       -0.267064052501
Balance.to.1layhid3       0.340572635591
NumOfProducts.to.1layhid3 -1.664740359564
HasCrCard.to.1layhid3     1.039052993587
IsActiveMember.to.1layhid3 0.150309306242
```

Type here to search

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins

Untitled1* | z | infert | Untitled2* | College | Untitled3* | iris | cars | airquality | Untitled4* | Bank1 | Untitled5*

Source on Save | Run | Source

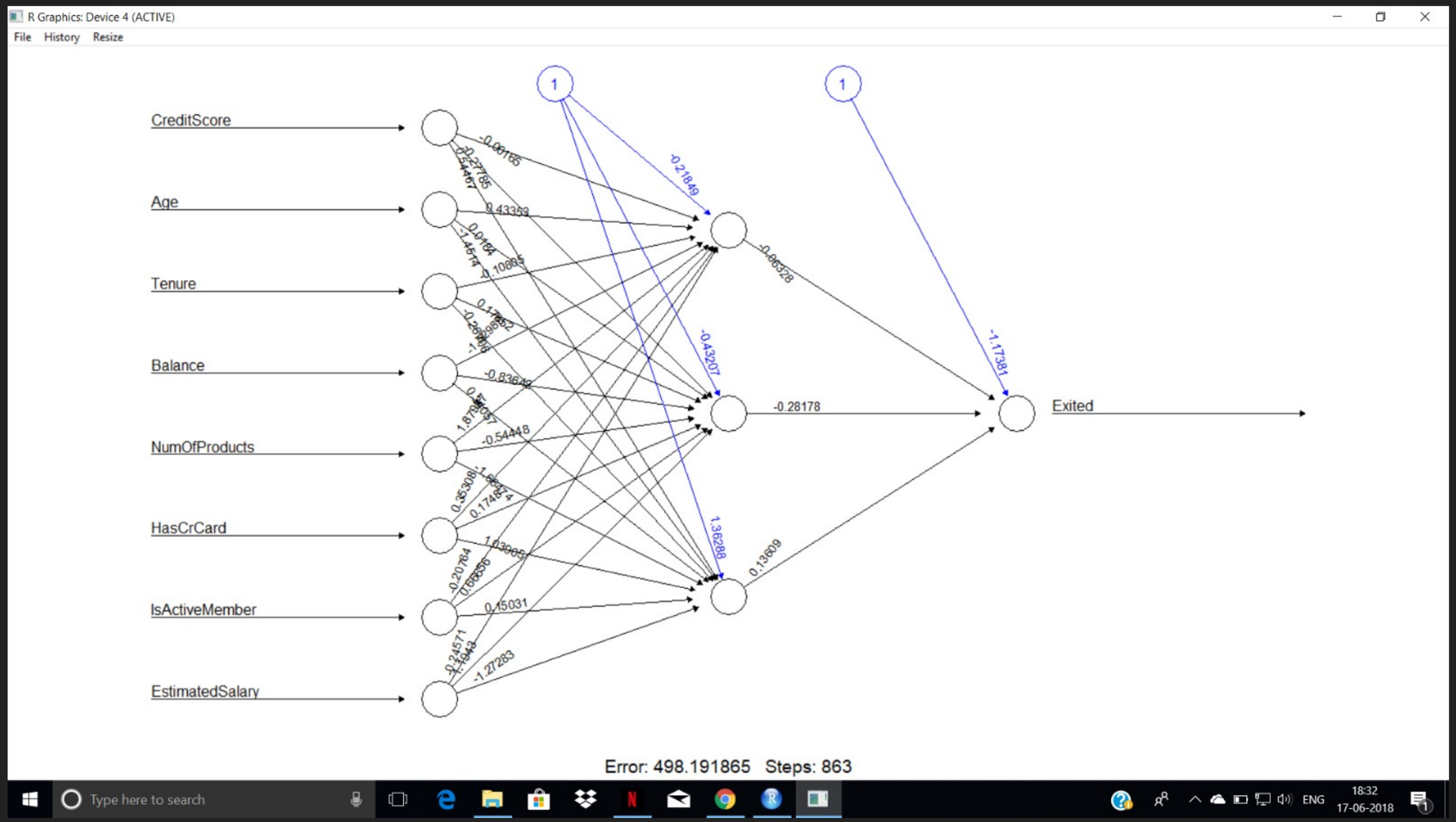
```
5                data = Bank1 , hidden = 3 , learningrate = 0.001   , algorithm = "backprop" , err.fct = "ce" ,
6                linear.output = FALSE)
7
8   nn.bp
9
10  plot(nn.bp)
11
12  nn.bp$net.result
13
14  nn.bp$weights
15
16  nn.bp$covariate
17
18  Bank1$Exited
19
20  nn.bp$net.result[[1]]
21
22  nn1=ifelse(nn.bp$net.result[[1]]>0.2 , 1, 0)
23  nn1
24
25  misClasificationError = mean(Bank1$Exited != nn1)
26  misClasificationError
27
```

27:1 | Traditional Backpropagation Algorithm | R Script

Console | Terminal

```
961    0
962    0
963    1
964    0
965    0
966    1
967    0
968    0
969    0
970    0
971    1
972    0
973    0
974    0
975    0
976    1
977    1
978    0
979    1
980    0
981    0
982    0
983    0
984    1
985    1
986    0
987    1
> misClasificationError = mean(Bank1$Exited != nn1)
> misClasificationError
[1] 0.3424518744
>
```

Type here to search


R Graphics: Device 4 (ACTIVE)
File History Resize
CreditScore
Age
Tenure
Balance
NumOfProducts
HasCrCard
IsActiveMember
EstimatedSalary
1
1
-0.21849
-0.43207
-0.06328
-0.28178
-1.17381
1.36288
0.13609
Exited
Error: 498.191865 Steps: 863
Type here to search
ENG
18:32
17-06-2018

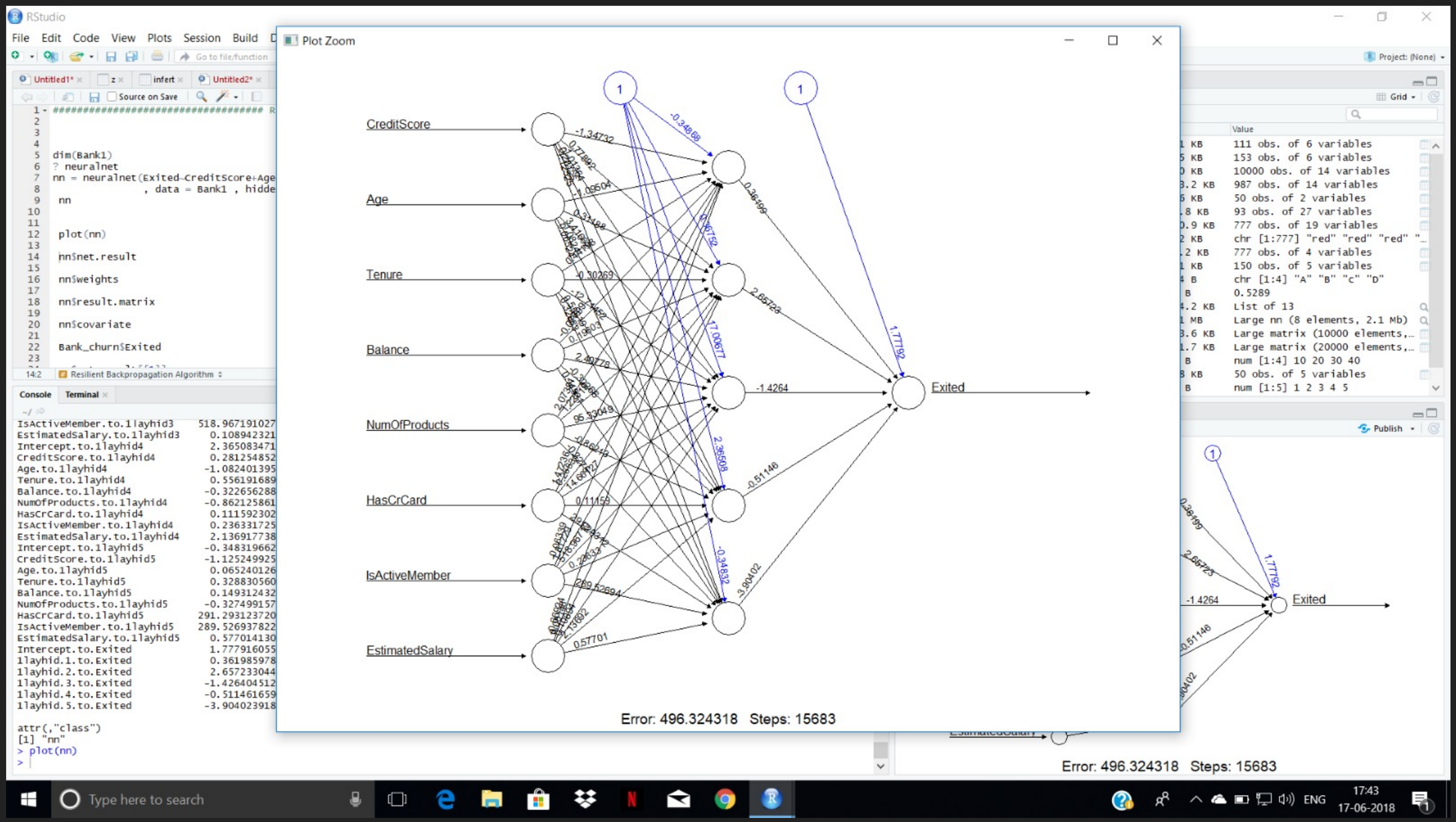
3. RESILIENT BACK PROPAGATION :

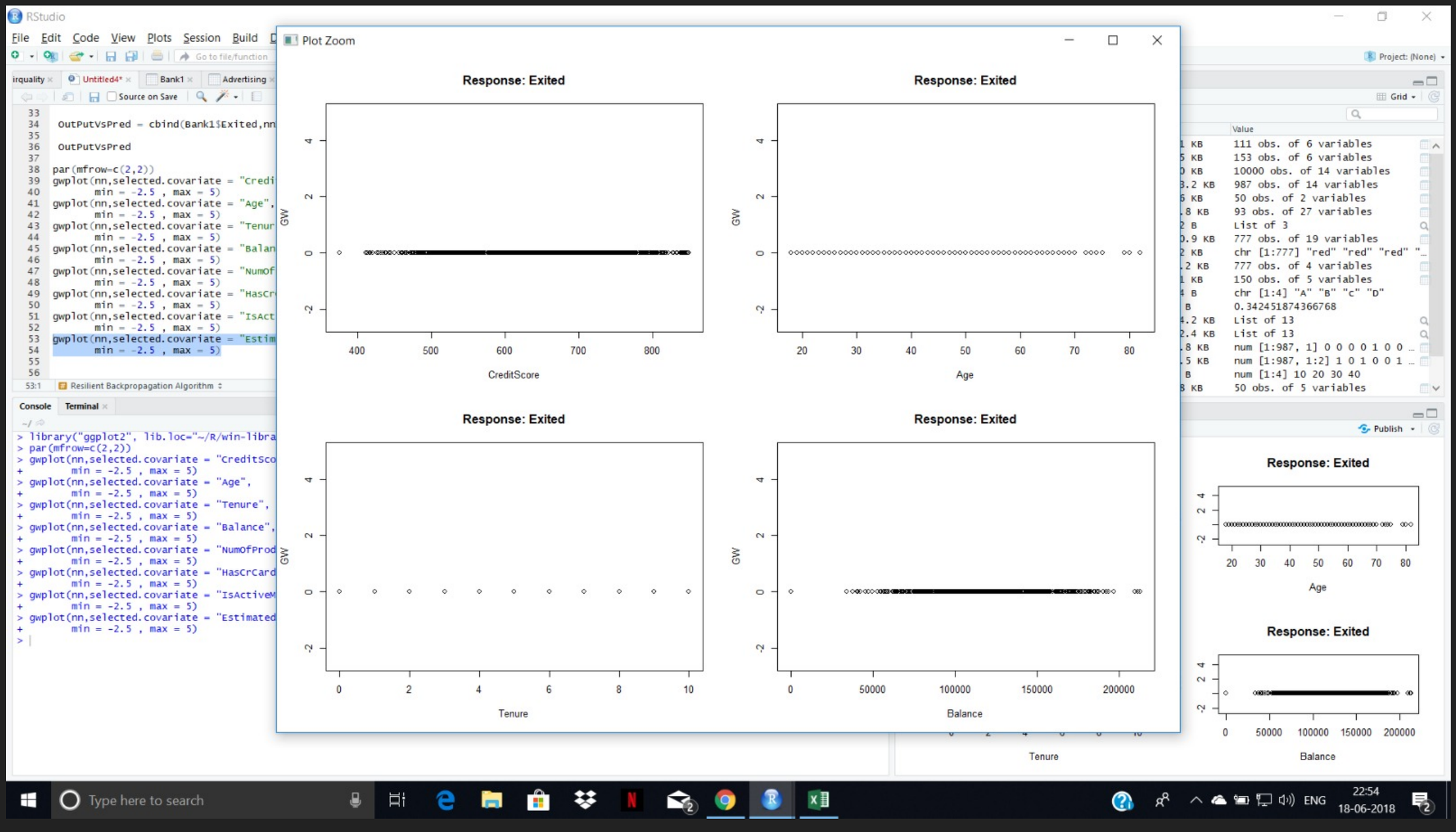
- Resilient back propagation (Rprop), an algorithm that can be used to train a neural network, is similar to the more common (regular) back-propagation.

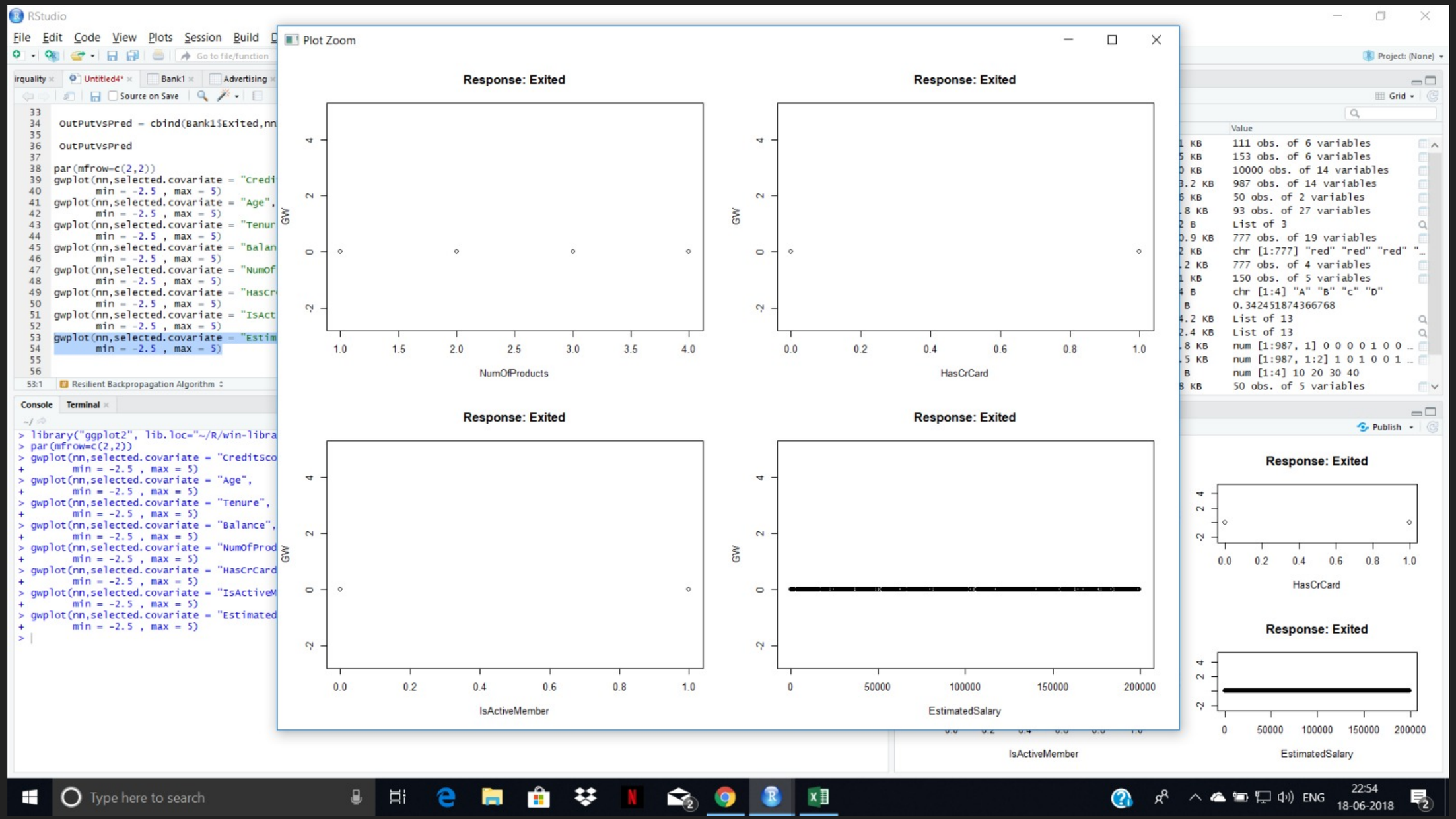
- But it has two main advantages over back propagation:

  - First, training with Rprop is often faster than training with back propagation.

  - Second, Rprop doesn't require you to specify any free parameter values, as opposed to back propagation which needs values for the learning rate (and usually an optional momentum term).


RStudio
File Edit Code View Plots Session Build
Plot Zoom
CreditScore
Age
Tenure
Balance
NumOfProducts
HasCrCard
IsActiveMember
EstimatedSalary
Exited
Error: 496.324318 Steps: 15683
Resilient Backpropagation Algorithm
Console Terminal
> plot(nn)
Type here to search
ENG
17:43
17-06-2018

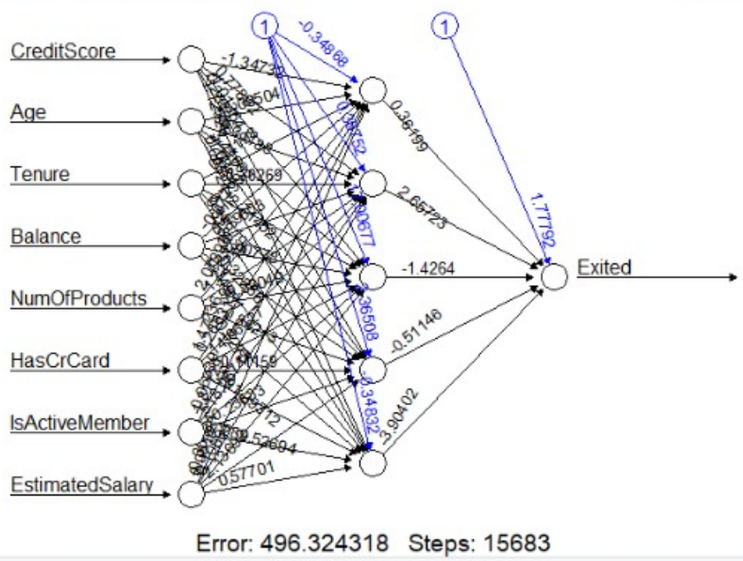

RStudio
File Edit Code View Plots Session Build
Plot Zoom
Response: Exited
GW
CreditScore
Age
Tenure
Balance
Console
Terminal
Resilient Backpropagation Algorithm
Type here to search
ENG
22:54
18-06-2018


RStudio
File Edit Code View Plots Session Build
Plot Zoom
Response: Exited
GW
NumOfProducts
HasCrCard
IsActiveMember
EstimatedSalary
Console
Terminal
Publish
Type here to search
ENG
22:54
18-06-2018

```
nn$weights

nn$result.matrix

nn$covariate

Bank1$Exited

nn$net.result[[1]]

nn1=ifelse(nn$net.result[[1]]>0.2 , 1, 0)

nn1

misClasificationError = mean(Bank1$Exited != nn1)

misClasificationError

OutPutVsPred = cbind(Bank_churn$Exited,nn1)

OutPutVsPred
```

```
961    0
962    0
963    0
964    0
965    0
966    0
967    0
968    0
969    0
970    0
971    0
972    0
973    0
974    0
975    0
976    1
977    0
978    0
979    0
980    0
981    0
982    0
983    0
984    0
985    1
986    0
987    0
> misClasificationError = mean(Bank1$Exited != nn1)
> misClasificationError
[1] 0.2299898683
>
```

Name	Type	Length	Size	Value
a1	data.frame	6	8.1 KB	111 obs. of 6 variables
airquality	data.frame	6	5.5 KB	153 obs. of 6 variables
Bank_churn	data.frame	14	820 KB	10000 obs. of 14 variables
Bank1	data.frame	14	113.2 KB	987 obs. of 14 variables
cars	data.frame	2	1.6 KB	50 obs. of 2 variables
Cars93	data.frame	27	35.8 KB	93 obs. of 27 variables
College	data.frame	19	130.9 KB	777 obs. of 19 variables
color	character	777	6.2 KB	chr [1:777] "red" "red" "red" "…
dt	data.frame	4	13.2 KB	777 obs. of 4 variables
iris	data.frame	5	7.1 KB	150 obs. of 5 variables
labels	character	4	304 B	chr [1:4] "A" "B" "c" "D"
misClasificationE…	numeric	1	56 B	0.229989868287741
nn	nn	13	464.2 KB	List of 13
nn.bp	nn	8	2.1 MB	Large nn (8 elements, 2.1 Mb)
nn1	matrix	987	69.8 KB	num [1:987, 1] 0 0 0 0 0 0 0 …
OutPutVsPred	matrix	20000	781.7 KB	Large matrix (20000 elements,…
slices	numeric	4	80 B	num [1:4] 10 20 30 40
vRows	data.frame	5	3.8 KB	50 obs. of 5 variables
x	numeric	5	96 B	num [1:5] 1 2 3 4 5



1. NAIVE BAYES :

- The Naive Bayes Classifier technique is based on the so-called Bayesian theorem.
- The model is very naively believing that the effect of the occurrence of any of the events is completely independent of the occurrence of other events.
- Naive Bayes is known to provide results at par and sometimes even better than highly complex and computationally expensive classification models.
- Advantages of Naive Bayes:
  - Super simple, you're just doing a bunch of counts.
  - You need less training data.

NAIVE BAYES CLASSIFIER USING PYTHON :

jupyter Untitled1 Last Checkpoint: 37 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

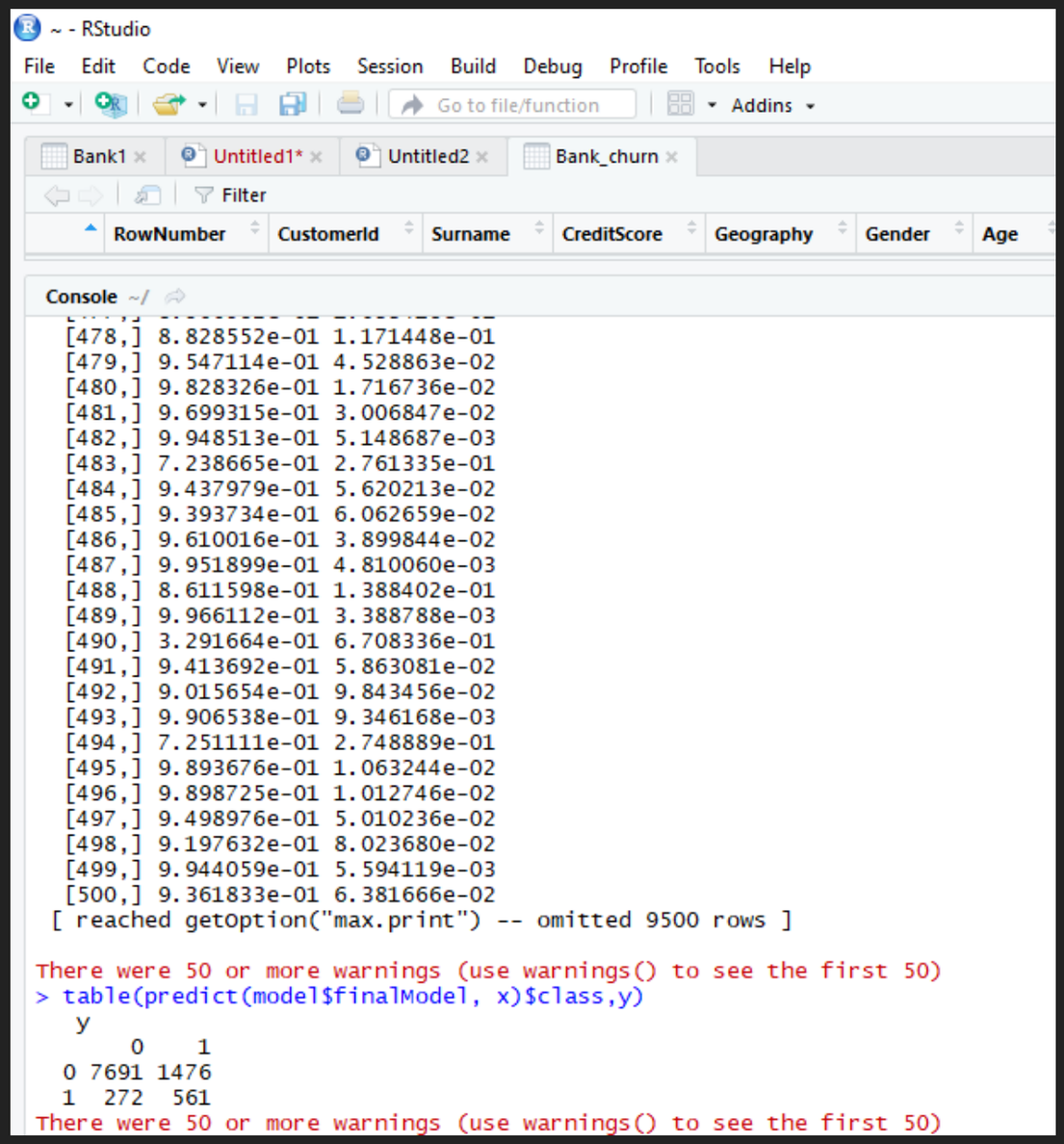
Trusted | Python 3

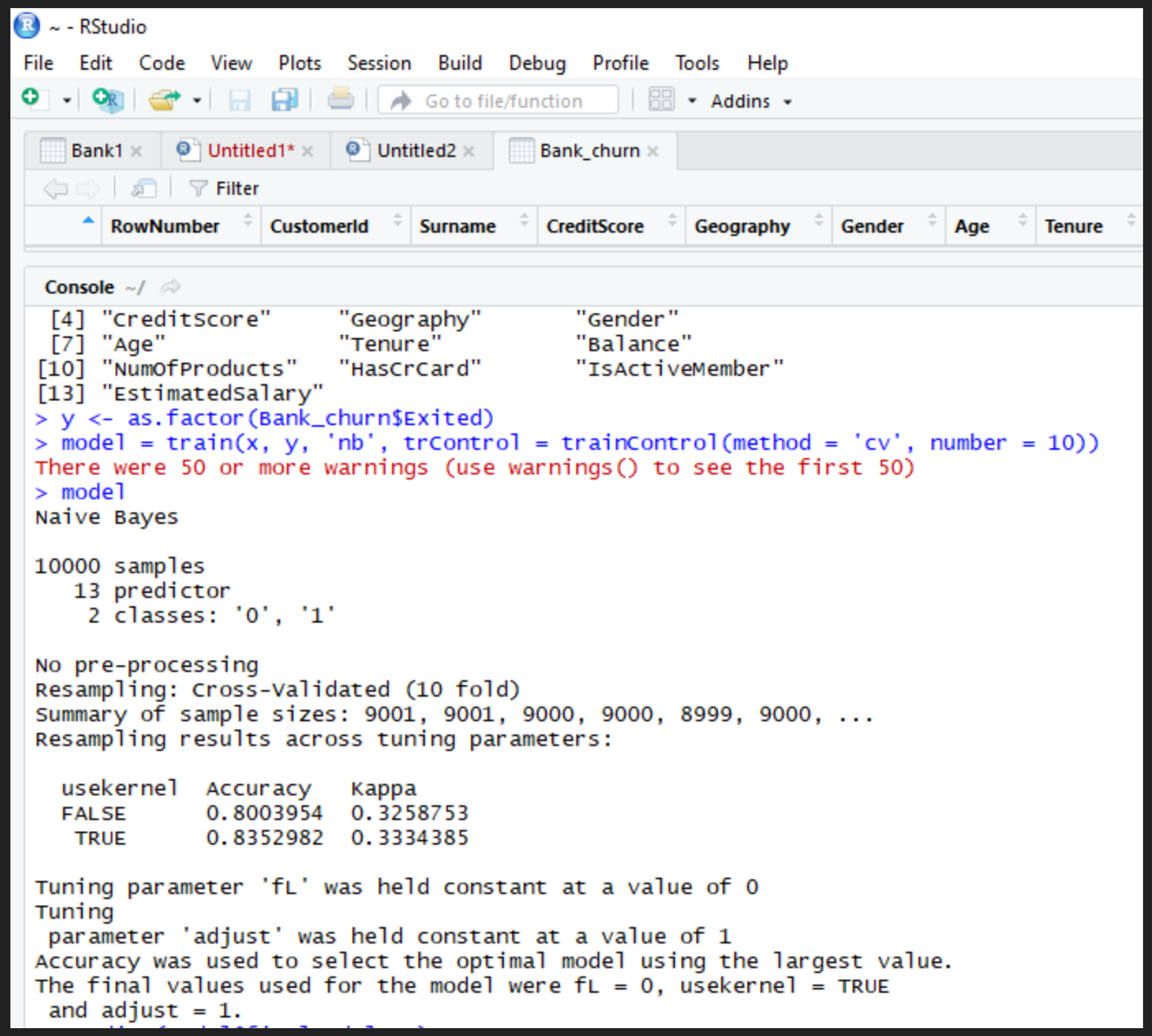
Run Code

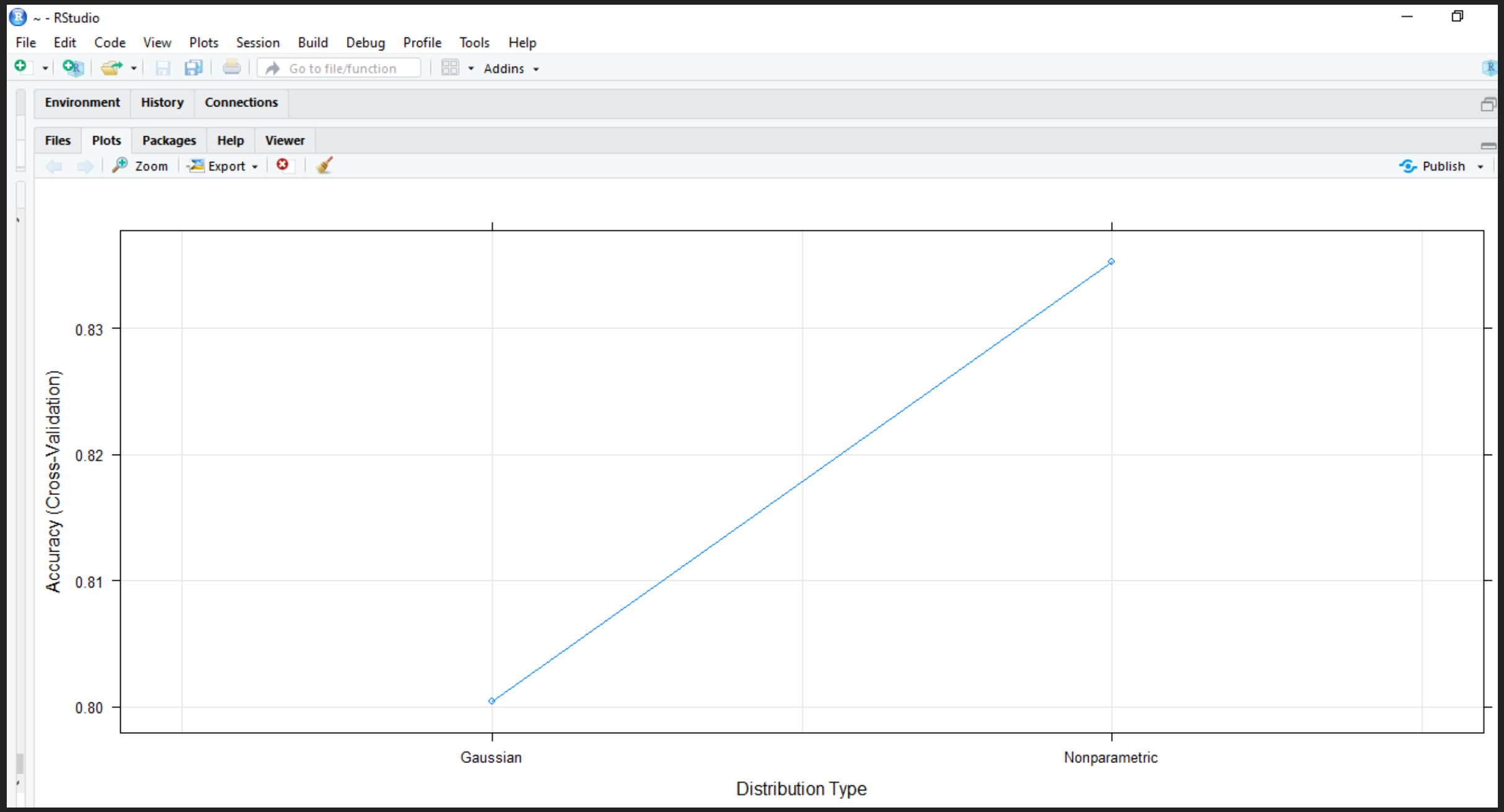
```
df = df.apply(le.fit_transform)
X = df.iloc[:,2:11]
Y = df.iloc[:,12]
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.33, random_state=42)
model = GaussianNB()
y_pred=model.fit(X_train,Y_train)
predicted= model.predict(X_test)
model.predict(X_test)
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(Y_test,predicted)
print(cm)
from sklearn.metrics import accuracy_score
print('Accuracy score',accuracy_score(Y_test, predicted))
from sklearn.metrics import classification_report
print(classification_report(Y_test, predicted))
```

```
[[2604   53]
 [ 487  156]]
Accuracy score 0.8363636363636363
             precision    recall  f1-score   support

          0       0.84      0.98      0.91      2657
          1       0.75      0.24      0.37       643

avg / total       0.82      0.84      0.80      3300
```

NAIVE BAYES CLASSIFIER USING R :






~ - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Environment History Connections
Files Plots Packages Help Viewer
Zoom
Export
Publish
0.83
0.82
0.81
0.80
Accuracy (Cross-Validation)
Gaussian
Nonparametric
Distribution Type

2. – STOCHASTIC GRADIENT DESCEND ALGORITHM (ADAM)

- Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data.

- Adam is different to classical stochastic gradient descent.

- Stochastic gradient descent maintains a single learning rate (termed alpha) for all weight updates and the learning rate does not change during training.

- Adam realizes the benefits of both AdaGrad and RMSProp.

y_pred - NumPy array

	0
0	False
1	False
2	False
3	False
4	False
5	True
6	False
7	False
8	False
9	True
10	False
11	False
12	False
13	False
14	True
15	True
16	False
17	False
18	False
19	False
20	True
21	False
22	False

Format Resize Background color

Type here to search

y_pred - NumPy array

	0
0	0.272587
1	0.281927
2	0.203548
3	0.0648228
4	0.0621671
5	0.838686
6	0.0111173
7	0.122228
8	0.171976
9	0.804126
10	0.0180612
11	0.145782
12	0.340869
13	0.245431
14	0.763506
15	0.654012
16	0.0982911
17	0.0507906
18	0.0727544
19	0.0825576
20	0.577071
21	0.00303745
22	0.0449607

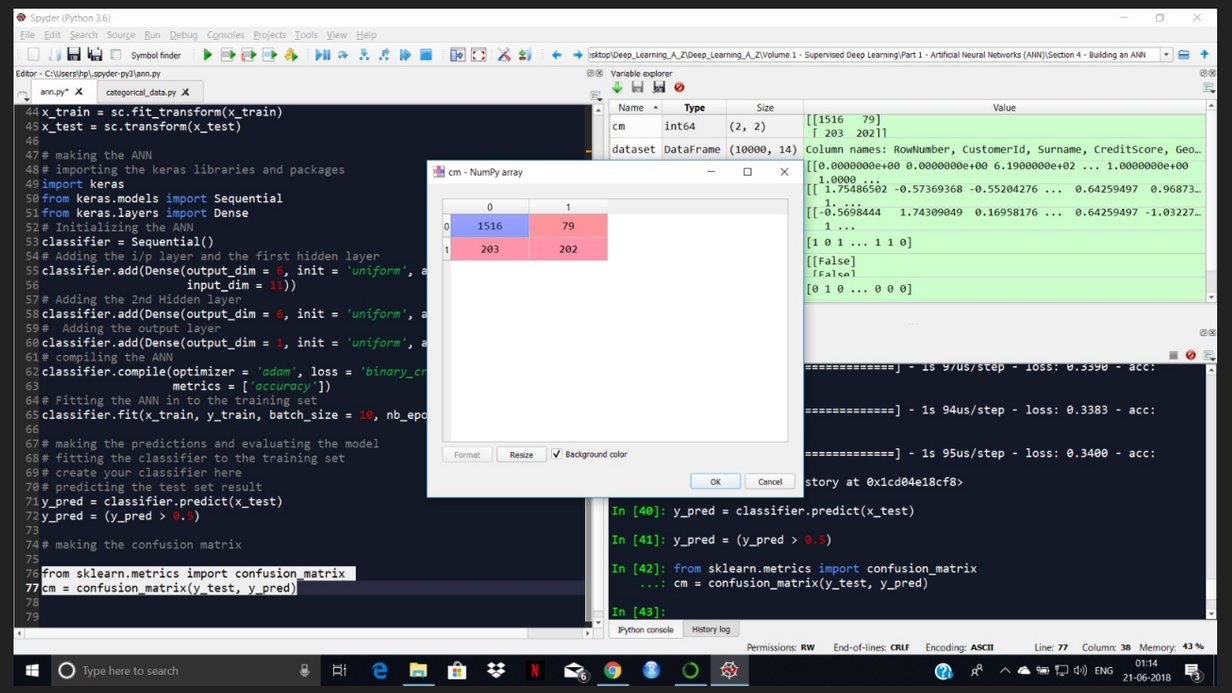
Format Resize Background color

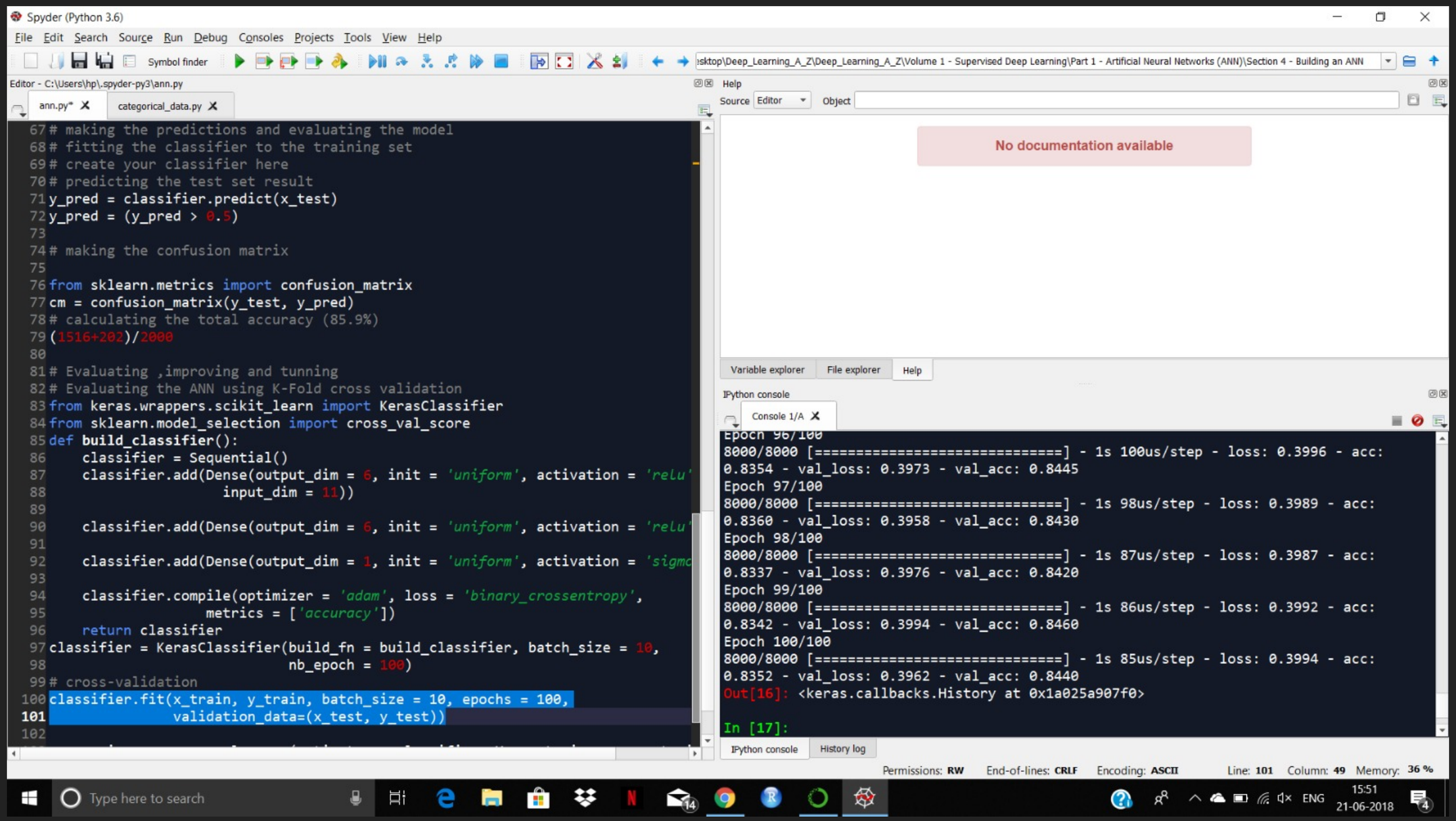
Type here to search

```
from keras.layers import Dense
# Initializing the ANN
classifier = Sequential()
# Adding the i/p layer and the first hidden layer
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu',
                     input_dim = 11))
# Adding the 2nd Hidden layer
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu'))
#  Adding the output layer
classifier.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmoid'))
# compiling the ANN
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy',
                   metrics = ['accuracy'])
# Fitting the ANN in to the training set
classifier.fit(x_train, y_train, batch_size = 50, nb_epoch = 150) (# accuracy=

# making the predictions and evaluating the model
# fitting the classifier to the training set
# create your classifier here
# predicting the test set result
y_pred = classifier.predict(x_test)
y_pred = (y_pred > 0.5)

# making the confusion matrix for test set

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
# calculating the total accuracy (85.9%)
(1516+202)/2000

# Evaluating ,improving and tunning
# Evaluating the ANN using K-Fold cross validation
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import cross_val_score
def build_classifier():
    classifier = Sequential()
```

Name	Type	Size	Value
dataset	DataFrame	(10000, 14)	Column names: RowNumber, CustomerId, Surname, CreditScore, Geogr…
x	float64	(10000, 11)	[[0.0000000e+00 0.0000000e+00 6.1900000e+02 ... 1.0000000e+00 1.0000 ...
x_test	float64	(2000, 11)	[[1.75486502 -0.57369368 -0.55204276 ... 0.64259497 0.9687384 1. ...
x_train	float64	(8000, 11)	[[-0.5698444 1.74309049 0.16958176 ... 0.64259497 -1.03227043 1 ...
y	int64	(10000,)	[1 0 1 ... 1 1 0]
y_pred	bool	(2000, 1)	[[False] [False]
y_test	int64	(2000,)	[0 1 0 ... 0 0 0]
y_train	int64	(8000,)	[0 0 0 ... 0 0 1]

```
Epoch 147/150
8000/8000 [==============================] - 0s 16us/step - loss: 0.3691 - acc:
0.8399
Epoch 148/150
8000/8000 [==============================] - 0s 16us/step - loss: 0.3685 - acc:
0.8407
Epoch 149/150
8000/8000 [==============================] - 0s 16us/step - loss: 0.3688 - acc:
0.8399
Epoch 150/150
8000/8000 [==============================] - 0s 16us/step - loss: 0.3689 - acc:
0.8411
Out[12]: <keras.callbacks.History at 0x1e80b252e48>

In [13]: y_pred = classifier.predict(x_test)
    ...: y_pred = (y_pred > 0.5)

In [14]:
```




Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
Symbol finder
sktop\Deep_Learning_A_Z\Deep_Learning_A_Z\Volume 1 - Supervised Deep Learning\Part 1 - Artificial Neural Networks (ANN)\Section 4 - Building an ANN
Editor - C:\Users\hp\.spyder-py3\ann.py
ann.py*
categorical_data.py
# making the predictions and evaluating the model
# fitting the classifier to the training set
# create your classifier here
# predicting the test set result
y_pred = classifier.predict(x_test)
y_pred = (y_pred > 0.5)
# making the confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
# calculating the total accuracy (85.9%)
(1516+202)/2000
# Evaluating ,improving and tunning
# Evaluating the ANN using K-Fold cross validation
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import cross_val_score
def build_classifier():
classifier = Sequential()
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu'
input_dim = 11))
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu'
classifier.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmo
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy',
metrics = ['accuracy'])
return classifier
classifier = KerasClassifier(build_fn = build_classifier, batch_size = 10,
nb_epoch = 100)
# cross-validation
classifier.fit(x_train, y_train, batch_size = 10, epochs = 100,
validation_data=(x_test, y_test))
Help
Source Editor Object
No documentation available
Variable explorer File explorer Help
IPython console
Console 1/A
Epoch 96/100
8000/8000 [==============================] - 1s 100us/step - loss: 0.3996 - acc:
0.8354 - val_loss: 0.3973 - val_acc: 0.8445
Epoch 97/100
8000/8000 [==============================] - 1s 98us/step - loss: 0.3989 - acc:
0.8360 - val_loss: 0.3958 - val_acc: 0.8430
Epoch 98/100
8000/8000 [==============================] - 1s 87us/step - loss: 0.3987 - acc:
0.8337 - val_loss: 0.3976 - val_acc: 0.8420
Epoch 99/100
8000/8000 [==============================] - 1s 86us/step - loss: 0.3992 - acc:
0.8342 - val_loss: 0.3994 - val_acc: 0.8460
Epoch 100/100
8000/8000 [==============================] - 1s 85us/step - loss: 0.3994 - acc:
0.8352 - val_loss: 0.3962 - val_acc: 0.8440
Out[16]: <keras.callbacks.History at 0x1a025a907f0>
In [17]:
IPython console
History log
Permissions: RW
End-of-lines: CRLF
Encoding: ASCII
Line: 101
Column: 49
Memory: 36 %
Type here to search
ENG
15:51
21-06-2018

DISCUSSION & CONCLUSION

- The most efficient and accurate result was given by ADAM optimiser with accuracy of 84.40%.
- Followed by Naive Bayes, which gave an accuracy of 83.63%.
- We didn’t get much precise results from Resilient Back Propagation and Back Propagation.

BUSINESS INSIGHT & POSSIBLE APPLICATION :

- Our model can help the banks reduce the attrition rate of the customers leaving.
- Since number of customers and account holders add up to be the major revenue generators, any model helping in increasing the number of customers would be better for the bank.
- Bank would be knowing where its services are lacking and therefore can take the required steps thus attracting more customers.
- More customers would lead to greater revenue.

NUS
National University
of Singapore

GROUP 15

BIG DATA ANALYTICS USING ANN