Deep Learning vs traditional Machine Learning algorithms used in Credit Card Fraud Detection

User Configuration Manual

Name: Sapna Gupta
Student Number: X14115824
Course: Msc Data Analytics
College: National College of Ireland
School of Computing

Supervisor : Jason Roche

1. Introduction

This manual lists all the technical details required to design a credit card fraud detection application using advanced data mining software. Output of this process is in the form of business analytic reports , charts which explain fraud patterns , fraud rings and well trained binary classification model which can find fraud from new unseen credit card transactions. Objective of this study is to compare performance of deep learning with other binary classification techniques like RF , GBM and GLM and this study also:

1. Investigate the use of new advanced data mining software to analyse credit card transactions to find fraud more efficiently.
2. Finds different types of data mining approaches to identify patterns and fraud rings in credit card transactions.
3. Tries to find solutions to the class imbalance problem and show how these solutions can be implemented through the use of new advanced data mining softwares like H2O.

This document is supplementary to the dissertation thesis report submitted. Functional and business concepts related to credit card fraud application are very well demonstrated in the thesis report.

Application Environment Setup

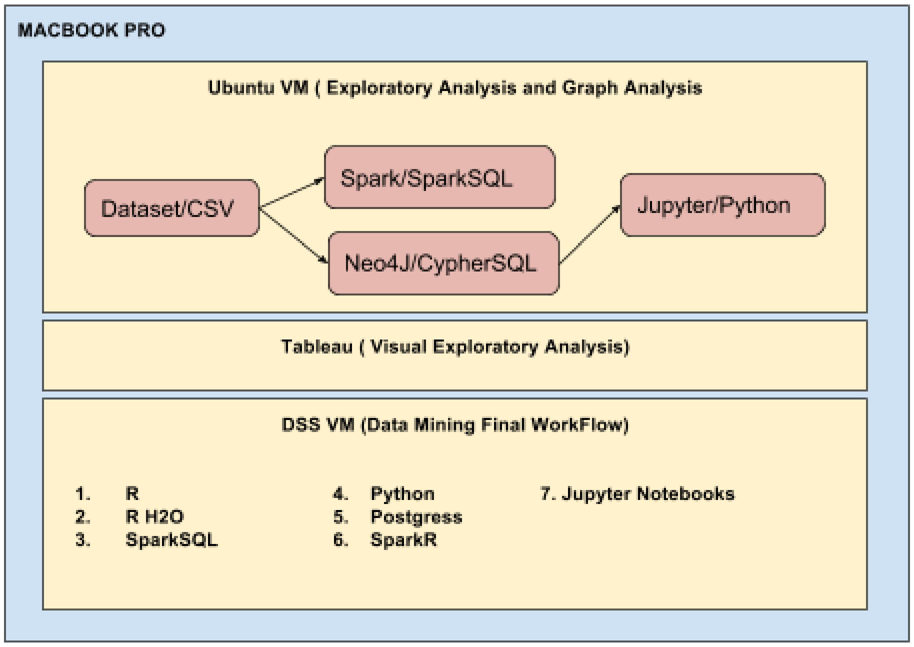


Figure 1. Overview of the hardware and software used during building process of the Fraud detection application

Hardware

1. Mac OS

Model Name:	MacBook Pro
Model Identifier:	MacBookPro11,5
Processor Name:	Intel Core i7
Processor Speed:	2.5 GHz
Number of Processors:	1
Total Number of Cores:	4
L2 Cache (per Core):	256 KB
L3 Cache:	6 MB
Memory:	16 GB

1.1 Ubuntu VM 14.04

Operating System Oracle (64-bit)
Base Memory : 11296 MB

All the advanced data mining softwares were manually loaded into this VM.

1.2 DSS Virtual Machine (VM) 3.0.0

Operating System Red Hat (64-bit)
Base Memory : 11856 MB

DSS VM is downloaded from Dataiku web site http://www.dataiku.com/dss/trynow/mac/ DSS comes with all the advanced data mining software preloaded. With the free version of DSS, user can only use limited softwares , so student licence was requested for using advance features or enterprise version of DSS. Due to easy installation of DSS and integration of all the advanced data mining softwares into DSS it became the best choice to carry out our final data modelling activity.

Software

Engineering tools like Spark , Neo4J , Python , Pyspark, R , SparkR , SparkSQL , Tableau , DSS and H2O were used to study the acquired dataset.

Apache **Spark** is advertised as "lightning fast cluster computing". Spark lets you run programs up to 100x faster in memory, or 10x faster on disk, than Hadoop. Spark contains very powerful and widely used libraries like SparkSQL, Spark Streaming, MLlib (for machine learning), and GraphX. Spark can be used with python, java and R through its APIs.[1]

Spark was downloaded into VM1 using the following link. Version of the spark used for the thesis is spark-1.5.2-bin-hadoop2.6.
http://spark.apache.org/downloads.html
Please follow all the steps provided in the below link to install scala and spark.
http://blog.prabeeshk.com/blog/2014/10/31/install-apache-spark-on-ubuntu-14-dot-04/

Neo4j is a one of the popular Graph Databases and uses Cypher Query Language. Neo4j is written in Java Language. It easily uncovers difficult-to-detect patterns as compared to relational databases.

Community edition of Neo4J graph database was downloaded into VM 1 from the following link. Neo4J community edition 2.3.3 was used for this thesis.
https://neo4j.com/download/
Please follow all the instructions provided in the below link to install Neo4J database.
https://www.digitalocean.com/community/tutorials/how-to-install-neo4j-on-an-ubuntu-vps

[1] "Apache Spark™ - Lightning-Fast Cluster Computing." 2014. 14 Aug. 2016 <http://spark.apache.org/>

Python is very powerful language and and Ipython Notebook is user interface tools widely used in data mining community and are easy to install and learn.

Python 2.7 was installed in VM1 using steps in the below link.
http://heliumhq.com/docs/installing_python_2.7.5_on_ubuntu
Ipython Notebook was installed in VM1 using instructions from the following link.
https://ipython.org/install.html

Tableau is a powerful BI tool for designing dashboards, it can be connected to any database , it has a very easy designer interface to create visualizations by just dragging and dropping. It is not an open source but it is made free to explore using student's licence.
Tableau enterprise edition was downloaded direct on the MACBOOK from the below link. Tableau 9.3 was used for this thesis.
http://www.tableau.com/products/
Student license was requested to explore full power of this tool.

Dataiku(Data Science Studio) DSS, is a complete solution to build any BI application. DSS can be used by a wide variety of people working as data professional like data scientists and data analysts. It gives an easy user interface to prototype, build, and deploy high quality services that transform raw data into effective business predictions.

DSS is not an open source tool but the enterprise edition is made free for researchers by using their student licence. DSS comes with most of the commonly softwares used in Data mining community. Which saved lots of time in building the pipelines and create workflows.
DSS enterprise edition VM was downloaded from the below link.
http://www.dataiku.com/dss/trynow/mac/
Student license was requested to explore full power of this tool.

H2O   is a built-in web interface for easy implementations such as GLM (linear regression, logistic regression, etc.), Naive Bayes, principal components analysis, time series, k-means clustering, Random Forest, Gradient Boosting and Deep Learning. Models can be build and can be trained with a very little knowledge of statistics. H2O can analyze billions of data rows in-memory, even with a small cluster. H2O is an open source and easy to install. H2O’s platform includes interfaces for R, Python, Scala, Java, JSON and many more.

For this project H2O is installed using following steps in R server in the DSS VM.

```
install.packages("h2o", repos=(c("http://s3.amazonaws.com/h2o-release/h2o/master/1497/R",
getOption("repos"))))
library(h2o)
localH2O = h2o.init()
```

Data Mining Process Details:

KDD (Knowledge Discovery in Databases) is commonly used methodology used in data mining applications. The process of finding hidden insights of the large data set is described in the following steps:

Dataset selection.

Dataset used for this thesis was acquired from UCSD (University of California, San Diego Data Mining Contest 2009). Dataset can be downloaded from [https://www.cs.purdue.edu/commugrate/data/credit_card/]

"The Dataset consists of two classification tasks based on e-commerce transaction anomaly data. The first task (task1)is to maximize accuracy of binary classification on a test data set, given a fully labeled training data set. The performance metric is the lift at 20% review rate. The second task (task2) is similar to task 1, but provides a couple of additional fields that have potential predictive information."[http://ebiquity.umbc.edu/blogger/2009/05/24/ucsd-data-mining-contest/]

Data Cleansing

Garbage in and garbage out, so the data cleaning preprocess is a very important step before starting any data mining project. It was found that there were no missing value in the dataset.

Preprocessing(using Spark)

Data set acquired from UCSD comprised of two sets: easy set (Task1) and a hard set (Task2).
Easy set consists of 2 data files: Task1Train and Task1Test.
Hard set consists of 2 data files: Task2Train and Task2Test.
These 4 input files were first analysed using SparkSQL for preprocessing.

1. Task 1 contains 17 continuous features and 2 categorical attributes (State and Domain_id).
    i. Task1Train contains 97,654 transactions.
    ii. Task1Test contains 30,000 transactions without class label.

2. Task 2 contains 17 continuous and 3 categorical attributes (State, customer_id and customer _email).
    i. Task2Train contains 100,000 transactions,
    ii. Task2Test contains 50,000 without class label.

Both these datasets were quited very different customer information missing from task1 and domain information missing from Task2. Task2 had contains additional fields with more predictive power , only Task2Train and Task2Test datasets were chosen for further analysis. Task2train had only 2692 fraudulent transactions, which is only 2.7% of the whole population.
(To rerun this analysis please refer to Scala code on Spark framework in Appendix Section A)

Exploratory Analysis (Using Tableau)

[Note: Tableau workbook with all the following charts is submitted as part of the proj. artifacts]

“Visualization is the act of making data understandable through the use of diagrams or pictorial methods to display complicated information or relationships held in data.” [http://www.dataminingmasters.com/uploads/studentProjects/thesis27v12P.pdf]

Load the data into tableau, using various plots relationship among various attributes is understood. Columns ‘Amount’ and ‘Total’ had same values. Columns ‘hour1’ and ‘hour2’ had same set of values. So we decided to drop Total and Hour2 columns from our final dataset.

Charts can be redwan using dragging and dropping of the columns as per the following Figures.

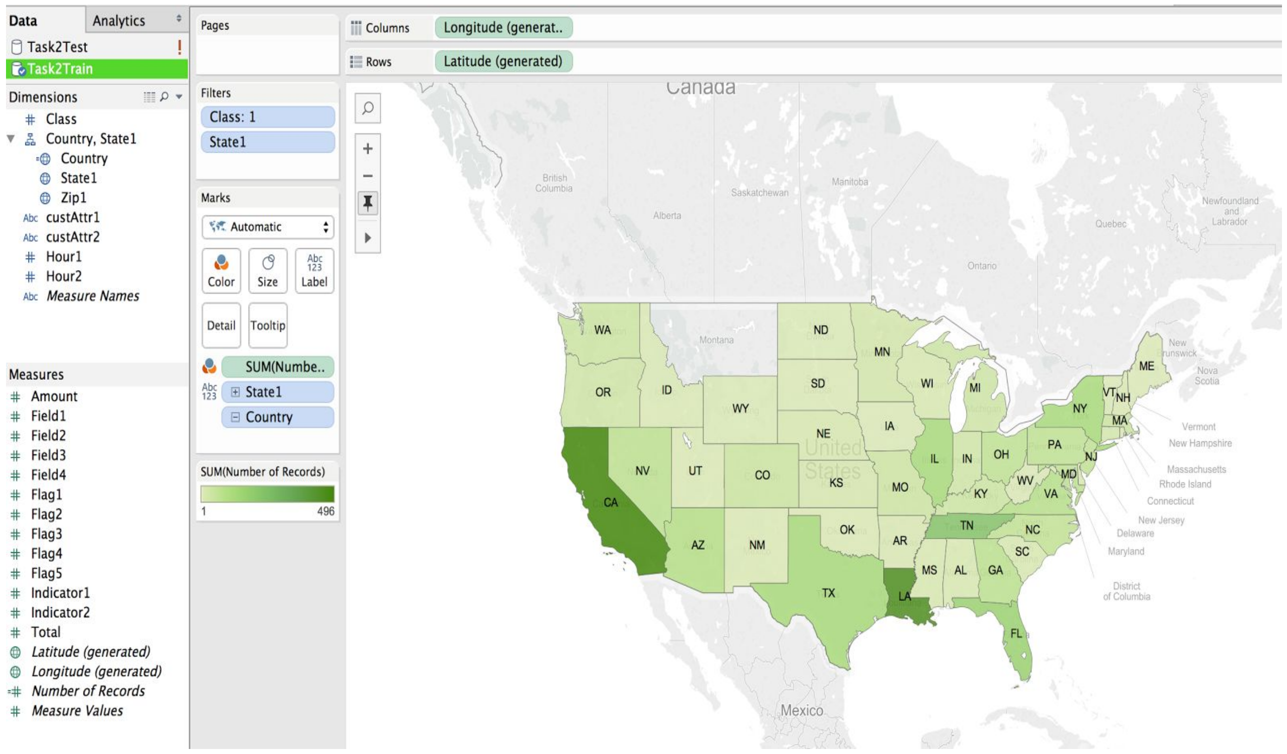


Figure 1: Tableau Report for finding US states from where most of the fraud transactions were initiated.

There are 54 states in the states column. Most transactions are recorded from the state “CA”.

A new calculated Country column was added. Column State and Zip code were changed into geological field types.

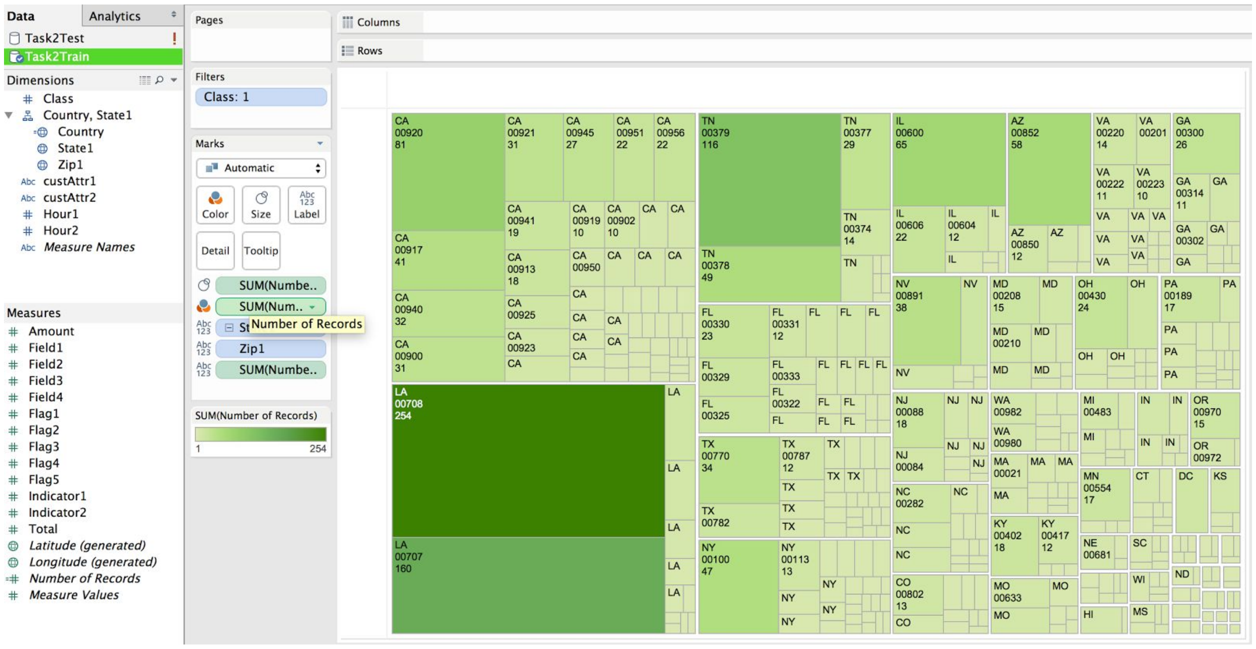


Figure 2: Tableau Report for finding zip code from where most of the fraud transactions were initiated.

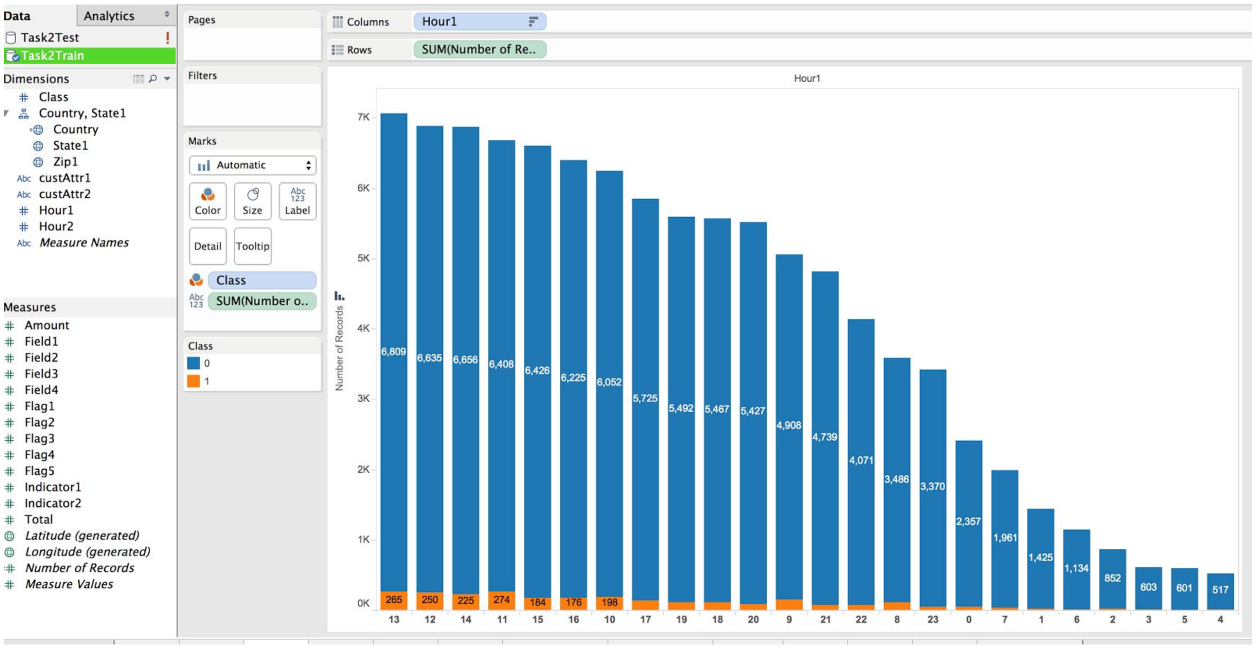


Figure 3: Tableau Report for finding hours of the day when most of the fraud transactions with initiated.

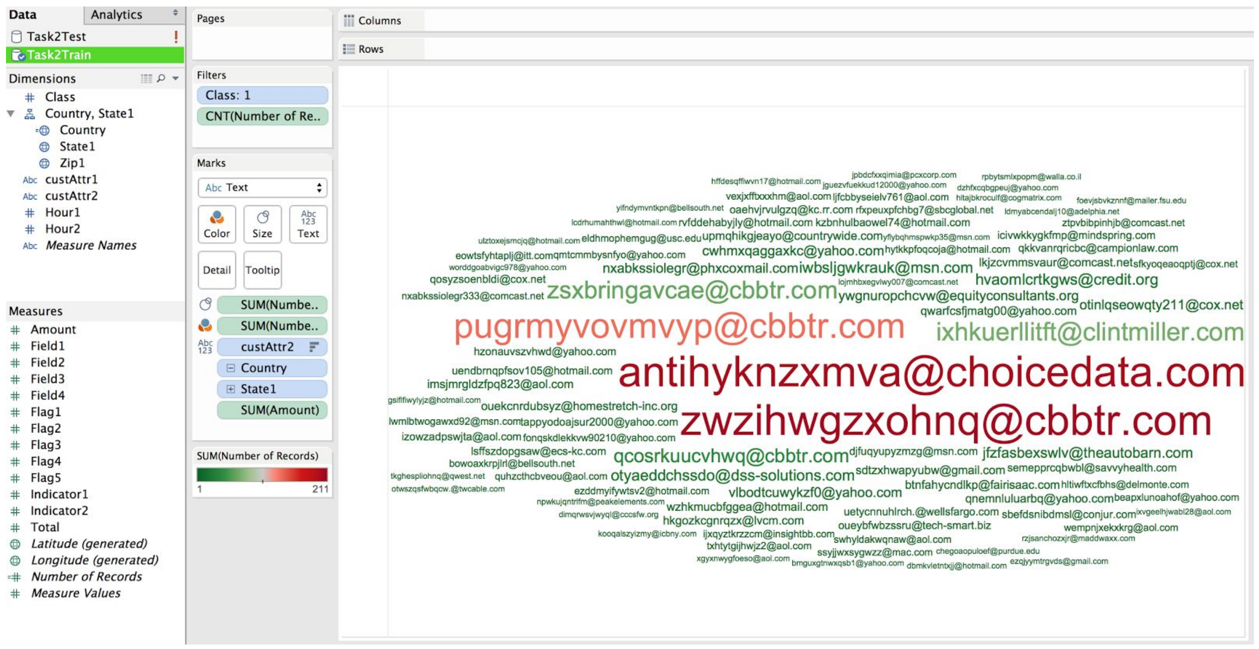


Figure 4: Tableau Report for finding email id used in most of the fraud transactions.

After creating individual reports , reports were integrated into a Tableau Dashboard like in the Figure 5.

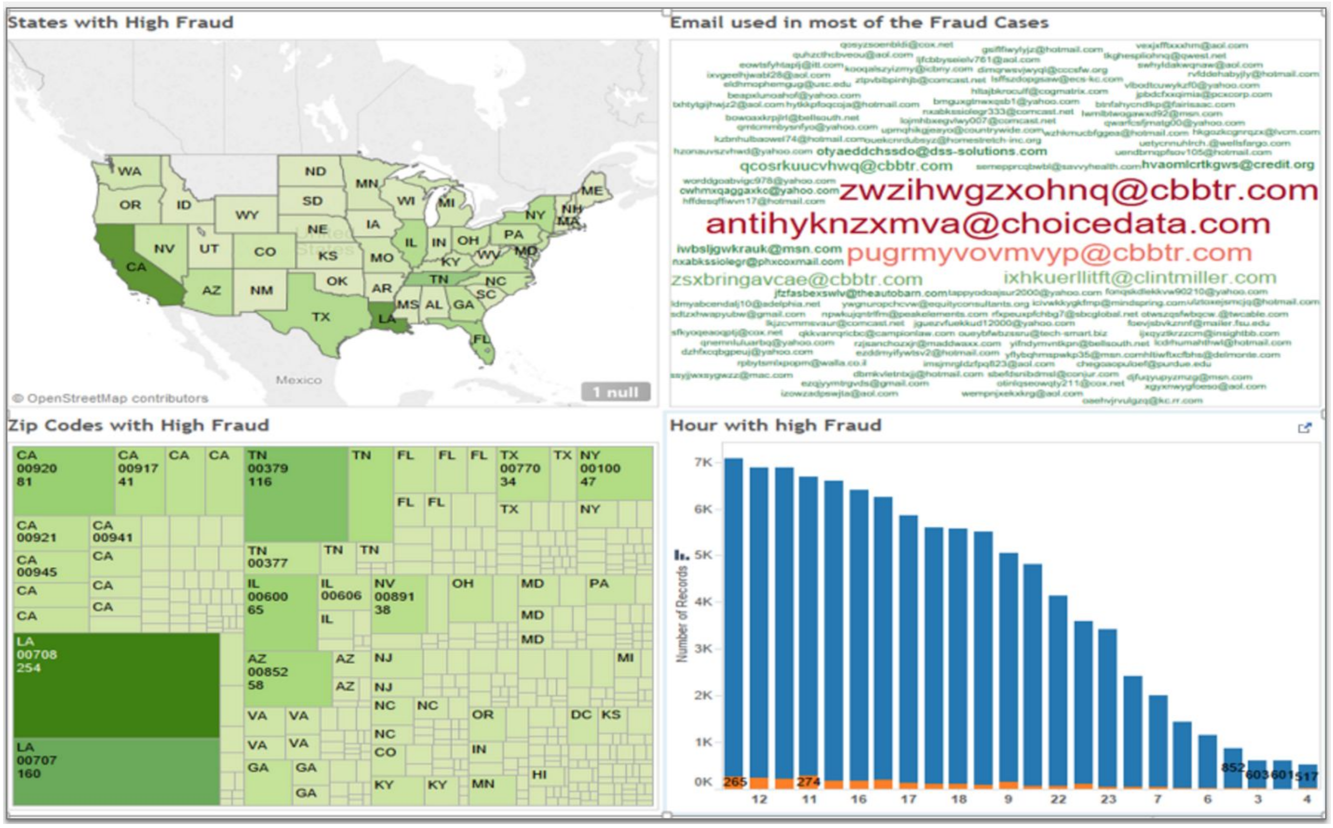
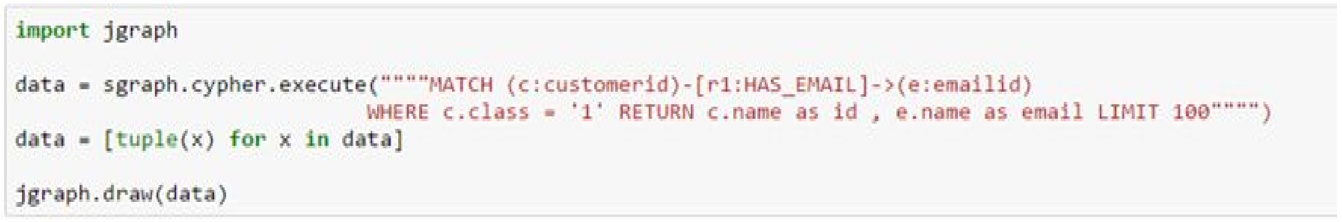


Figure 5: Tableau Dashboard for visual analysis of the dataset.

Social Network Analysis. (Using Neo4J0)

Social network analysis is study of social links or associations by using network theory which uses nodes and links. Every nodes in the network is one entity with the link being the association or links between other entities.
[http://www.dataminingmasters.com/uploads/studentProjects/thesis27v12P.pdf]

Load Task2Train dataset into Neo4J database.
For Creating Fraud database , creating Nodes and creating edges please follow the code step by step from Appendix Section B.
Then this dataset was queried from neo4j into ipython notebook to explore the fraud rings.For finding rings in the Neo4J database run the code from Appendix Section C in to ipython notebook. This code include implementation of 3D graph using python J-graph library to show fraud rings.



```
import jgraph

data = sgraph.cypher.execute("""MATCH (c:customerid)-[r1:HAS_EMAIL]->(e:emailid)
                                 WHERE c.class = '1' RETURN c.name as id , e.name as email LIMIT 100""")
data = [tuple(x) for x in data]

jgraph.draw(data)
```

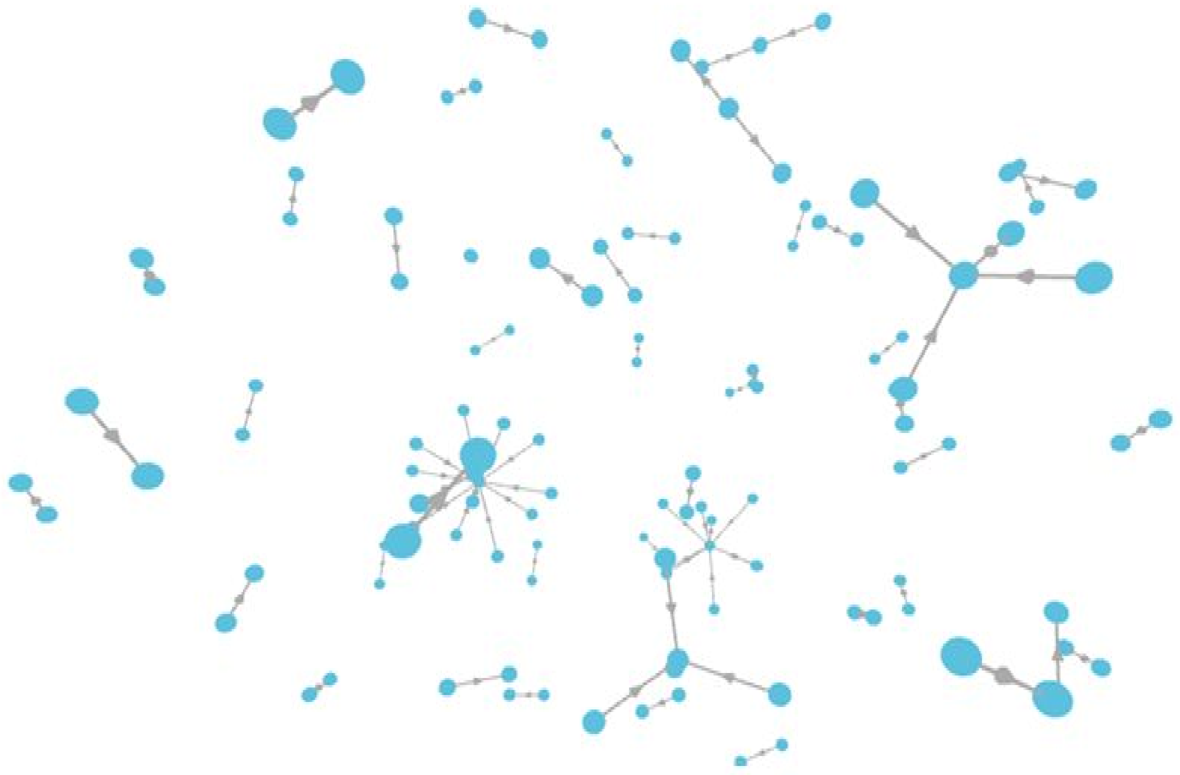


Figure 6: Showing Fraud rings found in the dataset using Neo4J and Jgraph library in ipython Notebook.

From the 3D graphs it was clear that most of the fraudsters used same email ids to commit fraud. There were over 100 fraudsters using the same email id, or it could be the fraudster, who created fake customer accounts using same email id. Conclusion of this graph analysis was that, email ids in Task2Train data set is a very interesting feature to learn fraudster’s behaviour. Tagging these emails could help to easily identify fraud in less time.

Data Sampling & Data Modelling

After exploratory analysis, dataset was loaded into DSS (Data Science Studio) for benchmarking different machine learning techniques. H2O was used in order to improve the training time of these models. Following are the workflows for each of the machine learning technique.

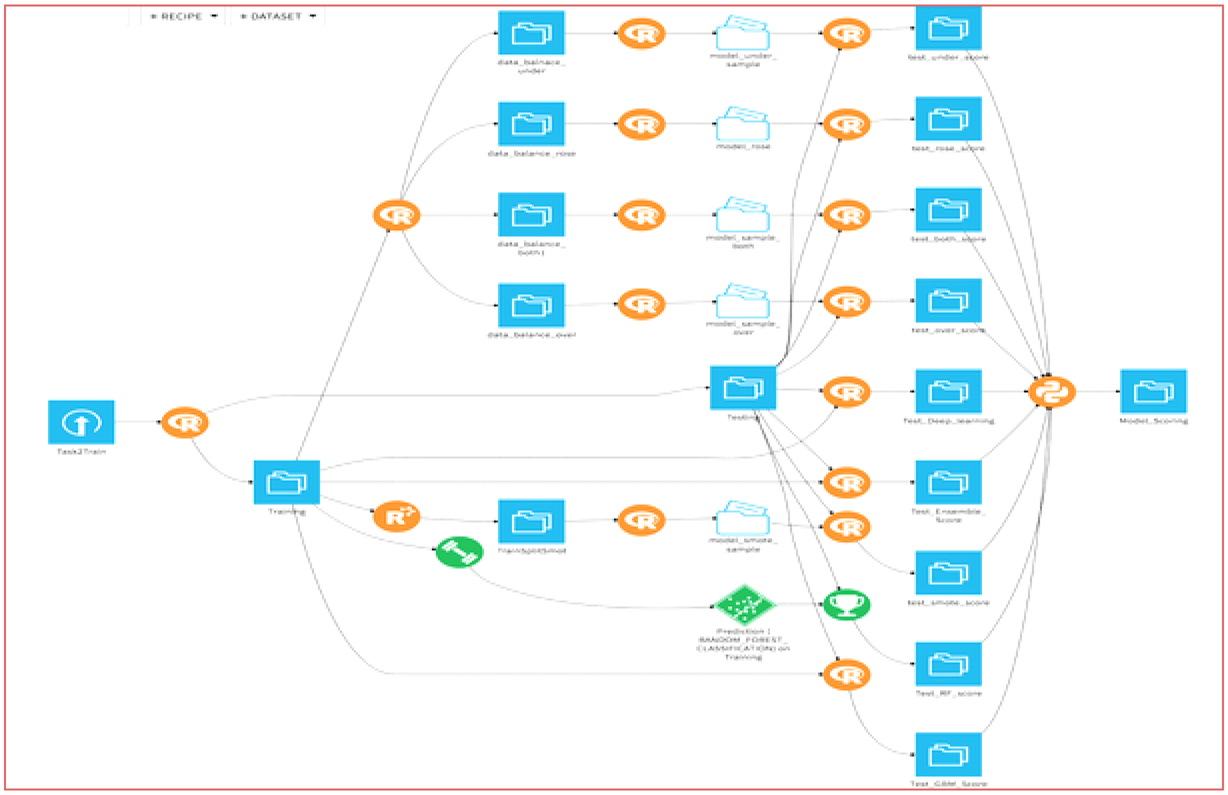
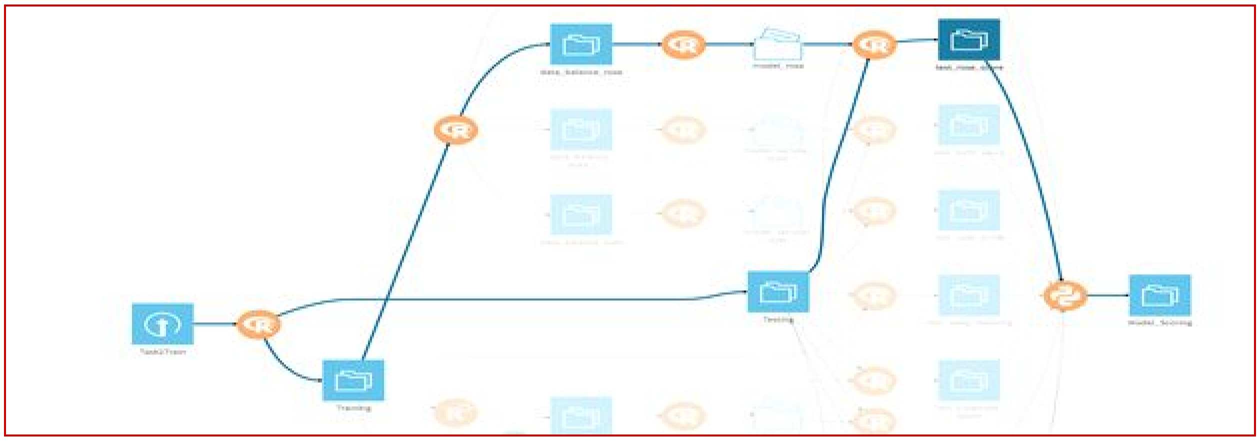


Figure 7: showing all the workflows created in DSS for benchmarking various data mining approaches used in finding fraud.

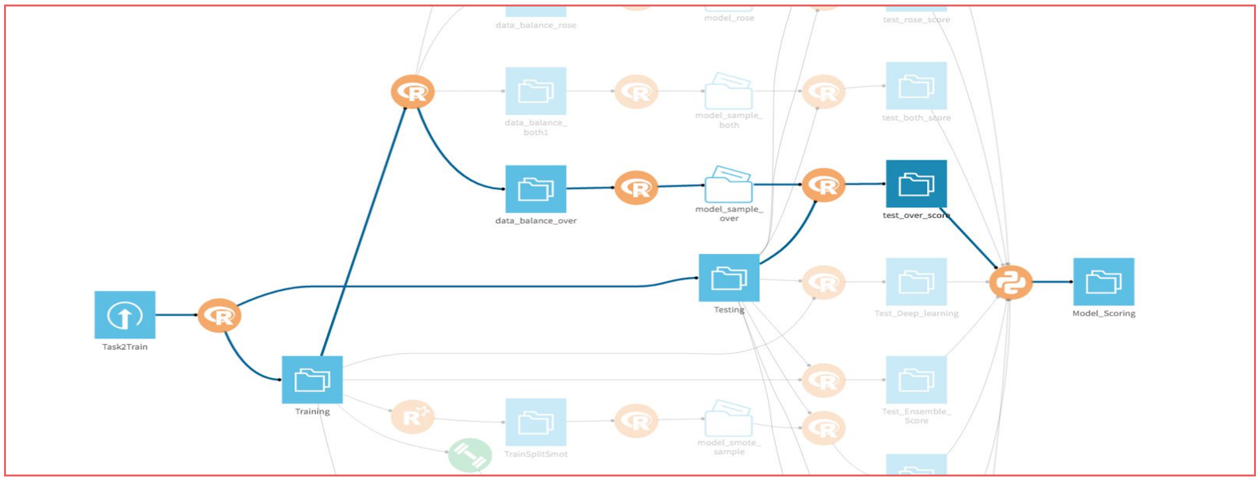
Below are common steps :

- Data is loaded into DSS
- Task2Train is split into 2 datasets ,70 percent for training and 30 percent for testing.(Please refer Appendix Section D for code details)



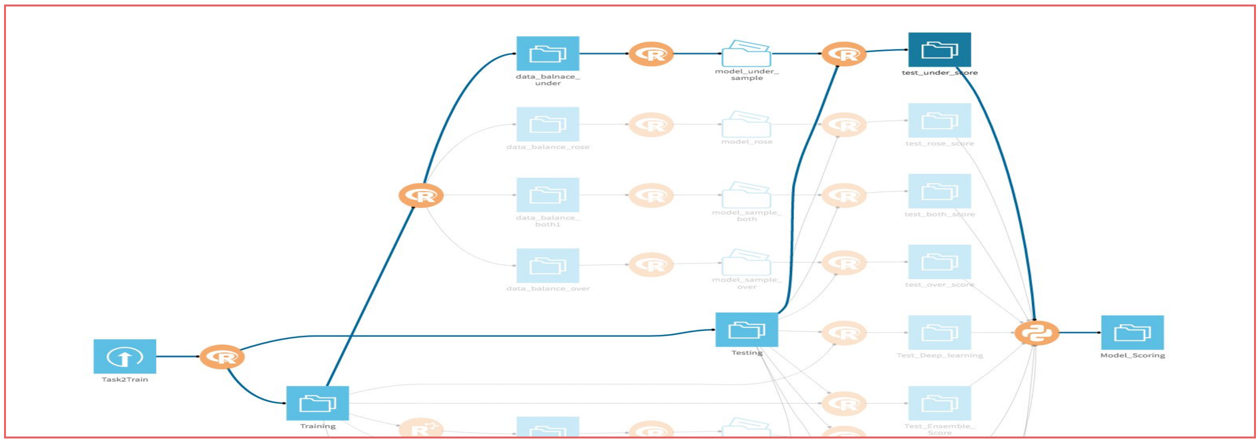
ROSE (Random Over Sampling)Analysis:

- ROSE Sampling is done on Training data set using ROSE function(Please refer Appendix Section E for code details)
- Rpart (Binary Classification decision tree) is used to train model (using ROSE sample , Please refer Appendix Section F)
- Train model is tested using testing dataset.(Please refer Appendix Section G)
- Results are analysed using Python.(Please refer Appendix Section N)



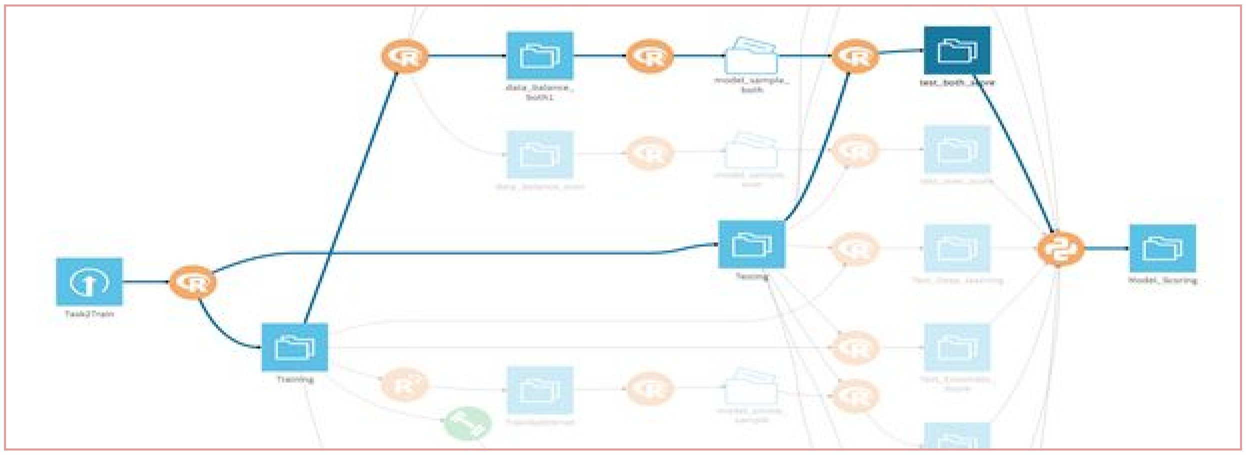
Over Sampling Analysis:

- Over Sampling is done on Training data set using ovun.sample function(Please refer Appendix Section E for code details)
- Rpart (Binary Classification decision tree) is used to train model (using hybrid sample , Please refer Appendix Section G)
- Train model is tested using testing dataset.(Please refer Appendix Section G)
- Results are analysed using Python.(Please refer Appendix Section N)



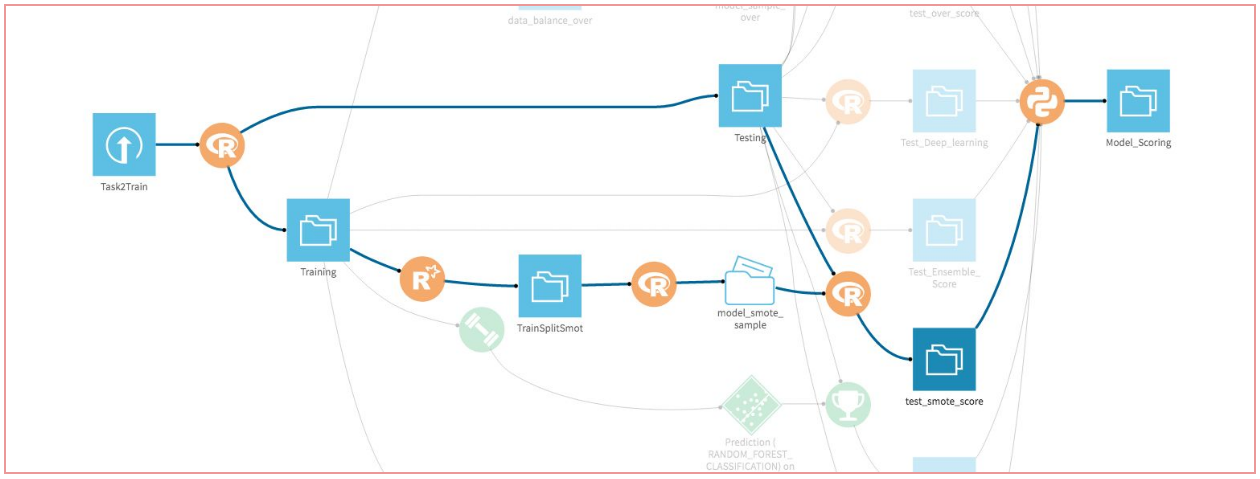
Under Sampling Analysis:

- Under Sampling is done on Training data set using ovun.sample function(Please refer Appendix Section E for code details)
- Rpart (Binary Classification decision tree) is used to train model (using hybrid sample , Please refer Appendix Section G)
- Train model is tested using testing dataset.(Please refer Appendix Section G)
- Results are analysed using Python.(Please refer Appendix Section N)



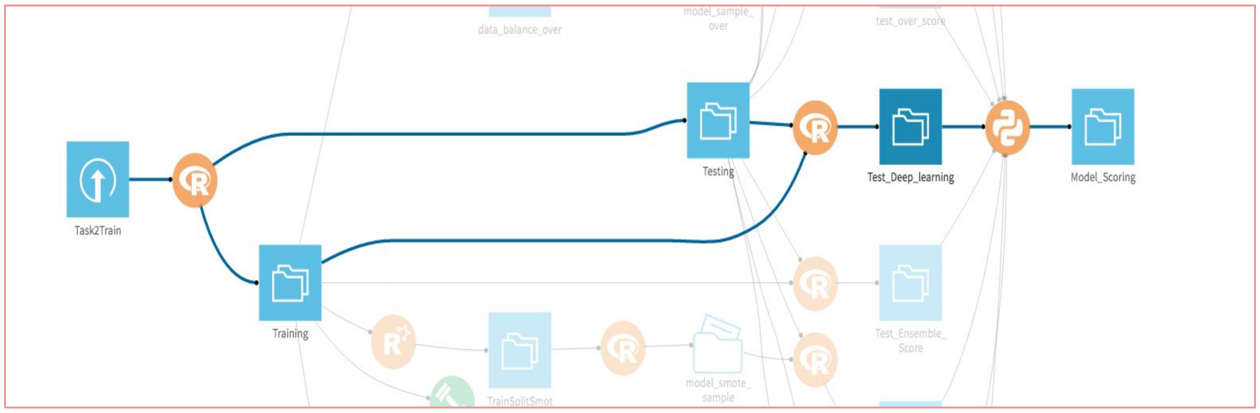
Hybrid (Over and Under Sampling)Analysis:

- Hybrid Sampling is done on Training data set using ovun.sample function(Please refer Appendix Section E for code details)
- Rpart (Binary Classification decision tree) is used to train model (using hybrid sample , Please refer Appendix Section G)
- Train model is tested using testing dataset.(Please refer Appendix Section G)
- Results are analysed using Python.(Please refer Appendix Section N)



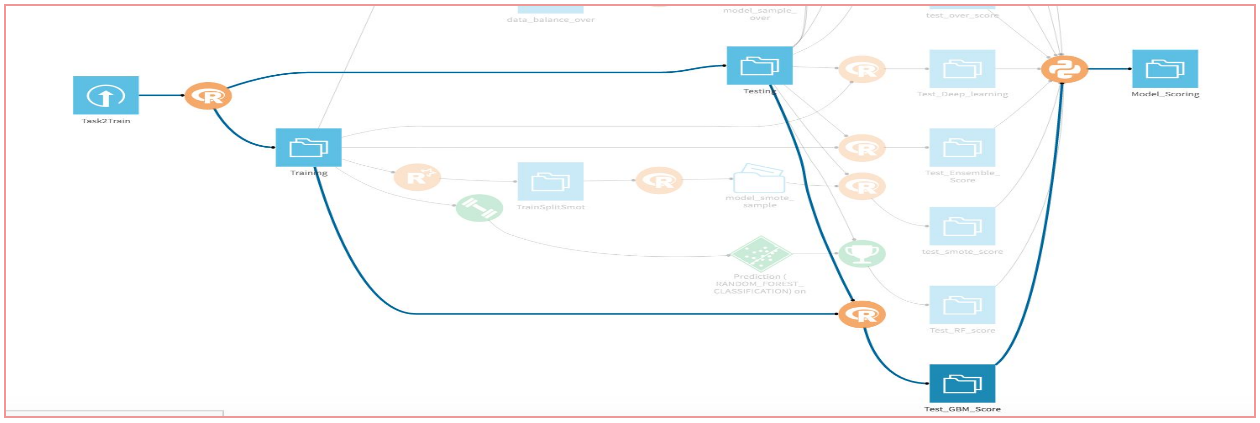
SMOTE Sampling Analysis:

- SMOTE Sampling is done on Training data set using SMOTE function is DMwR library in R.(Please refer Appendix Section E for code details)
- Rpart (Binary Classification decision tree) is used to train model (using smote sample , Please refer Appendix Section H)
- Train model is tested using testing dataset.(Please refer Appendix Section H)
- Results are analysed using Python.(Please refer Appendix Section N)



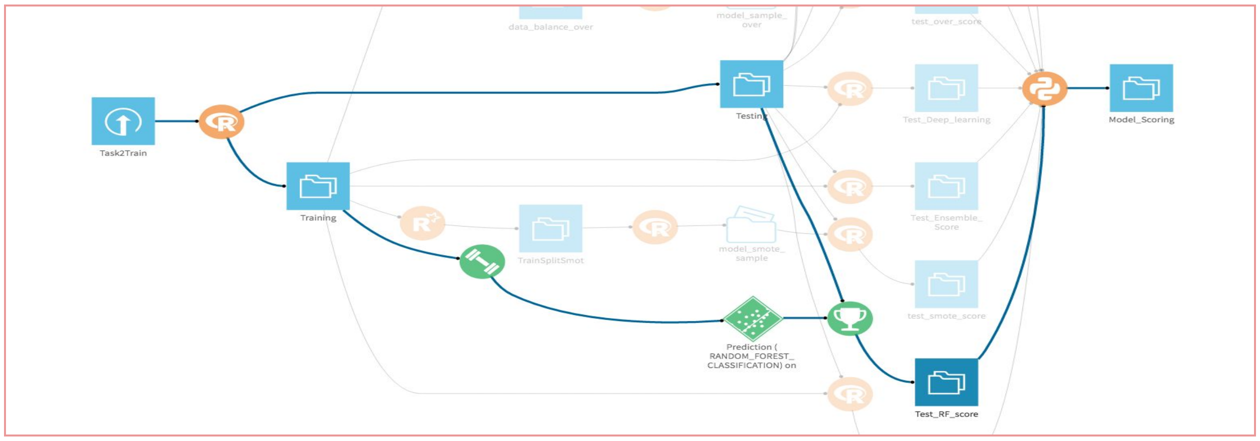
Deep Learning (neural network):

- Using Deep Learning method model is trained using training dataset (Please refer Appendix Section K)
- Trained model is applied on testing dataset to get predicted scores (Please refer Appendix Section K)
- Results are analysed using Python.(Please refer Appendix Section N)



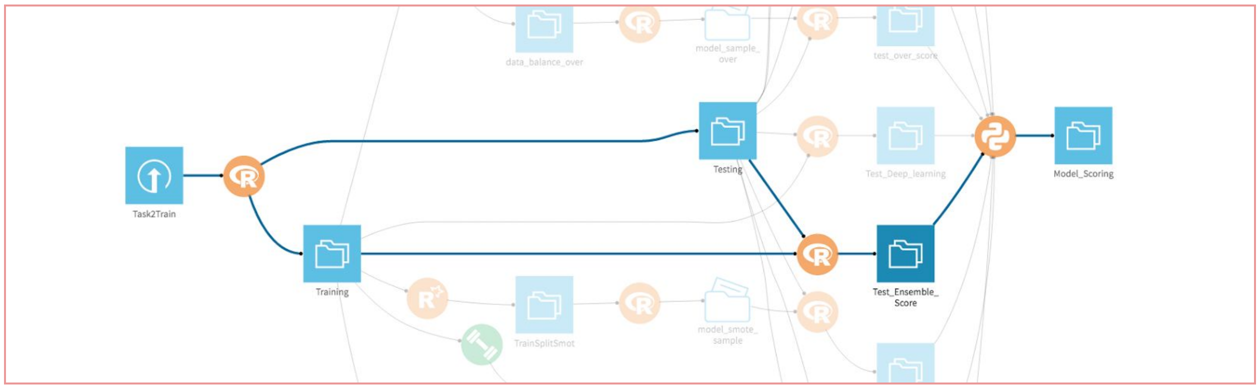
GBM Boosting Ensemble Learning :

- Using GBM ensemble method model is trained using training dataset (Please refer Appendix Section I)
- Trained model is applied on testing dataset to get predicted scores (Please refer Appendix Section I)
- Results are analysed using Python.(Please refer Appendix Section N)



RF Bagging Ensemble Learning :

- Using RF ensemble method model is trained using training dataset (there is no code for this , model was training using inbuilt DSS modelling functionality)
- Parameters select are --
- Trained model is applied on testing dataset to get predicted scores
- Results are analysed using Python.(Please refer Appendix Section N)



Stacking Ensemble Learning :

- Using stacking ensemble method model is trained using training dataset (Please refer Appendix Section J)
- Algorithms selected for stacking are GLM, RF, GBM , Deep Learning
- Trained model is applied on testing dataset to get predicted scores (Please refer Appendix Section J)

Deep Learning model tuning

Deep Learning model tuning using 5 fold cross validation.

Model trained using default parameters were compared with model trained using cross validation. Parameters used for cross validation as follows:

```
h2o.deeplearning(
                        x=1:17,
                        y=18,
                        training_frame="h2otrain",
                        hidden=c(50,50,50,50,50,50),
                        epochs=500,
                        nfolds=5,
                        balance_classes = TRUE,
                        fold_assignment="Stratified"
                       )
```

- (Please refer Appendix Section L)

Deep Learning model tuning hyper parameter tuning Grid Search.

Model trained using default parameters was compared with model trained using grid search.

Following is the list of parameters selected for grid search.

```
hyper_params <- list(
activation=c("Rectifier","Tanh","Maxout","RectifierWithDropout","TanhWithDropou
t","MaxoutWithDropout"),
  hidden=list(c(20,20),c(50,50),c(30,30,30),c(25,25,25,25),c(50,50,50,50,50,50)),
  epochs=500,
  input_dropout_ratio=c(0,0.05),
  l1=seq(0,1e-4,1e-6),
  l2=seq(0,1e-4,1e-6)
)
```

- (Please refer Appendix Section M)

Model no	model_ids	epochs	l2	l1	input_dropout_ratio	hidden	activation	logloss
1	dl_grid_random_model_4	500	0.000004	0.000045	0.05	25,25,25,25	Maxout	0.7989345929
2	dl_grid_random_model_2	500	0.000045	0.000029	0	30,30,30	Tanh	1.000089482
3	dl_grid_random_model_3	500	0.000094	0.000036	0	25,25,25,25	Rectifier	1.035693499
4	dl_grid_random_model_1	500	0.00006	0.000087	0.05	50,50	Tanh	1.080510091
5	dl_grid_random_model_5	500	0.000003	0.000008	0.05	50,50,50,50,50,50	Tanh	1.198024979
6	dl_grid_random_model_0	500	0.000045	0.00007	0.05	30,30,30	TanhWithDropout	1.471085054

Result Evaluation

All predicted scores of all the models build above are fed into python script to generate a list of performance metrics in Table 1 and plotted as in Fig 1 using excel.

- Results are analysed using Python script using Pandas , numpy and SKlearn metrics.(Please refer Appendix Section N)

Table 1

Model	AUC	Recal	MCC	Precision	F Score
Sample Smote	0.8084738213	0.7083333333	0.3231141435	0.1781201849	0.284658951
Sample both	0.8015451658	0.6948529412	0.3158566505	0.1747303544	0.2792415661
Sample Rose	0.8180207527	0.6446078431	0.6517818375	0.6778350515	0.6608040201
Sample Over	0.8247871517	0.681372549	0.4872872751	0.3746630728	0.4834782609
Sample Under	0.8069701771	0.7071078431	0.3193463134	0.1750606796	0.2806420233
DSS RF	0.6955735213	0.3946078431	0.5396476553	0.7612293144	0.5197740113
H2O GLM	0.6084518511	0.2181372549	0.4192484575	0.8317757009	0.345631068
H2O Deep Learning	0.7262279089	0.6850490196	0.1711495581	0.07608547707	0.1369594512
H2O GBM	0.7216182114	0.4509803922	0.5175373885	0.6195286195	0.5219858156
H2O RF	0.7503335833	0.5183823529	0.4674712081	0.45	0.4817767654
H2O Ensemble	0.7737741067	0.5539215686	0.6173574557	0.7084639498	0.6217331499

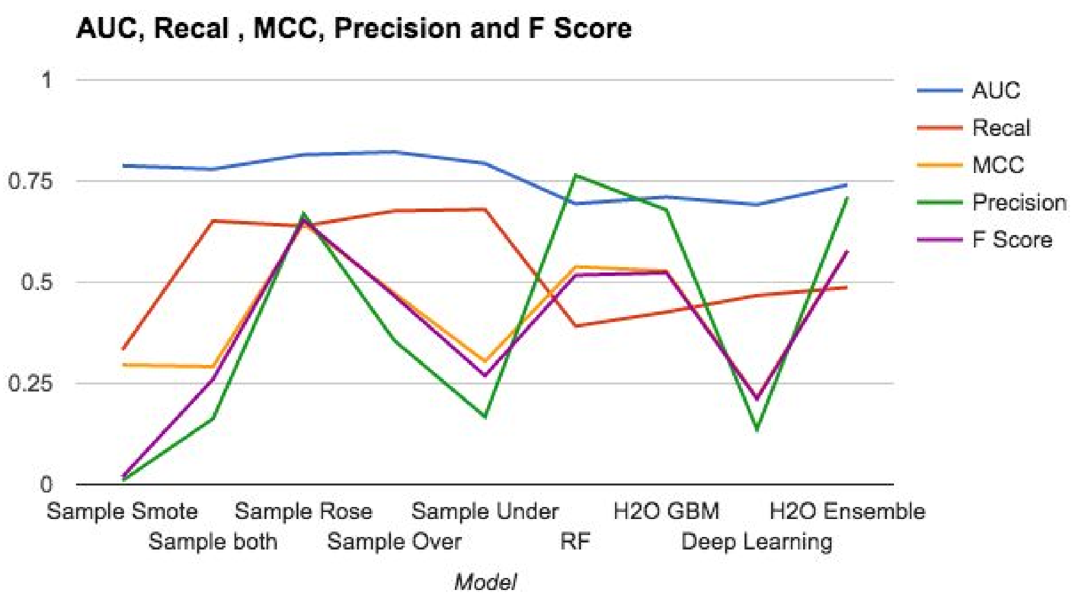


Fig 1 Results plotted in line chart (this chart is drawn using excel charting tool)

- Results are analysed using Python script using Pandas , numpy and SKlearn metrics.(Please refer Appendix Section N)

Appendix

Section A

```
val Task2Train = sc.textFile("/home/jroche/Downloads/Task2Train.csv")
val Task1Train = sc.textFile("/home/jroche/Downloads/Newtask1train.csv")
val Task2Test = sc.textFile("/home/jroche/Downloads/Task2Test.csv")
val Task1Test = sc.textFile("/home/jroche/Downloads/Task1Testcsv.csv")

case class Task2Train2 (amount:Double, hour1:Integer, state1:String, zip1:Integer, customer_id:String, field1:Integer,
email_id:String, field2:Integer, hour2:Integer,
flag1:Integer, total:Double, field3:Integer, field4:Integer, indicator1:Integer, indicator2:Integer, flag2:Integer, flag3:Integer,
flag4:Integer, flag5:Integer, Class_label:Integer )

case class Task2Test2 (amount:Double, hour1:Integer, state1:String, zip1:Integer, customer_id:String, field1:Integer,
email_id:String, field2:Integer, hour2:Integer,
flag1:Integer, total:Double, field3:Integer, field4:Integer, indicator1:Integer, indicator2:Integer, flag2:Integer, flag3:Integer,
flag4:Integer, flag5:Integer  )

case class Task1Train1 (amount:Double, hour1:Integer, state1:String, zip1:Integer, field1:Integer, domain_id:String, field2:Integer,
hour2:Integer,
flag1:Integer, total:Double, field3:Integer, field4:Integer, field5:Integer, indicator1:Integer, indicator2:Integer, flag2:Integer,
flag3:Integer, flag4:Integer, flag5:Integer, Class_label:Integer )
```

```
case class Task1Test1 (amount:Double, hour1:Integer, state1:String, zip1:Integer,  field1:Integer, domain_id:String, field2:Integer,
hour2:Integer,
flag1:Integer, total:Double, field3:Integer, field4:Integer, field5:Integer,indicator1:Integer, indicator2:Integer, flag2:Integer,
flag3:Integer, flag4:Integer, flag5:Integer )


//case class Train2Task2 (amount:Double, hour1:String, state1:String, zip1:String, custAttr1:String, field1:String, custAttr2:String,
field2:String, hour2:String,
//flag1:String, total:String, field3:String, field4:String, indicator1:String, indicator2:String, flag2:String, flag3:String, flag4:String,
flag5:String, Class_labels:String )

// split each line, filter out header (starts with "amount"), and map it into Train2Task2 case class

val Set2Train = Task2Train.map(s=>s.split(",")).filter(s=>s(0)!="amount").map(
  s=>Task2Train2(s(0).toDouble,
        s(1).replaceAll("\"", "").toInt,
        s(2).replaceAll("\"", ""),
        s(3).replaceAll("\"", "").toInt,
        s(4).replaceAll("\"", ""),
        s(5).replaceAll("\"", "").toInt,
        s(6).replaceAll("\"", ""),
        s(7).replaceAll("\"", "").toInt,
        s(8).replaceAll("\"", "").toInt,
        s(9).replaceAll("\"", "").toInt,
        s(10).replaceAll("\"", "").toDouble,
        s(11).replaceAll("\"", "").toInt,
        s(12).replaceAll("\"", "").toInt,
        s(13).replaceAll("\"", "").toInt,
        s(14).replaceAll("\"", "").toInt,
        s(15).replaceAll("\"", "").toInt,
        s(16).replaceAll("\"", "").toInt,
        s(17).replaceAll("\"", "").toInt,
        s(18).replaceAll("\"", "").toInt,
        s(19).replaceAll("\"", "").toInt
    )
)

val Set2Test = Task2Test.map(s=>s.split(",")).filter(s=>s(0)!="amount").map(
  s=>Task2Test2(s(0).toDouble,
        s(1).replaceAll("\"", "").toInt,
        s(2).replaceAll("\"", ""),
        s(3).replaceAll("\"", "").toInt,
        s(4).replaceAll("\"", ""),
        s(5).replaceAll("\"", "").toInt,
        s(6).replaceAll("\"", ""),
        s(7).replaceAll("\"", "").toInt,
        s(8).replaceAll("\"", "").toInt,
        s(9).replaceAll("\"", "").toInt,
        s(10).replaceAll("\"", "").toDouble,
        s(11).replaceAll("\"", "").toInt,
        s(12).replaceAll("\"", "").toInt,
        s(13).replaceAll("\"", "").toInt,
        s(14).replaceAll("\"", "").toInt,
        s(15).replaceAll("\"", "").toInt,
        s(16).replaceAll("\"", "").toInt,
        s(17).replaceAll("\"", "").toInt,
        s(18).replaceAll("\"", "").toInt
    )
)


val Set1Train = Task1Train.map(s=>s.split(",")).filter(s=>s(0)!="amount").map(
  s=>Task1Train1(s(0).toDouble,
        s(1).replaceAll("\"", "").toInt,
        s(2).replaceAll("\"", ""),
        s(3).replaceAll("\"", "").toInt,
        s(4).replaceAll("\"", "").toInt,
        s(5).replaceAll("\"", ""),
        s(6).replaceAll("\"", "").toInt,
        s(7).replaceAll("\"", "").toInt,
        s(8).replaceAll("\"", "").toInt,
        s(9).replaceAll("\"", "").toDouble,
        s(10).replaceAll("\"", "").toInt,
        s(11).replaceAll("\"", "").toInt,
        s(12).replaceAll("\"", "").toInt,
        s(13).replaceAll("\"", "").toInt,
        s(14).replaceAll("\"", "").toInt,
        s(15).replaceAll("\"", "").toInt,
        s(16).replaceAll("\"", "").toInt,
        s(17).replaceAll("\"", "").toInt,
        s(18).replaceAll("\"", "").toInt,
        s(19).replaceAll("\"", "").toInt
    )
)

val Set1Test = Task1Test.map(s=>s.split(",")).filter(s=>s(0)!="amount").map(
  s=>Task1Test1(s(0).toDouble,
        s(1).replaceAll("\"", "").toInt,
        s(2).replaceAll("\"", ""),
        s(3).replaceAll("\"", "").toInt,
        s(4).replaceAll("\"", "").toInt,
        s(5).replaceAll("\"", ""),
        s(6).replaceAll("\"", "").toInt,
        s(7).replaceAll("\"", "").toInt,
        s(8).replaceAll("\"", "").toInt,
        s(9).replaceAll("\"", "").toDouble,
        s(10).replaceAll("\"", "").toInt,
        s(11).replaceAll("\"", "").toInt,
        s(12).replaceAll("\"", "").toInt,
        s(13).replaceAll("\"", "").toInt,
        s(14).replaceAll("\"", "").toInt,
        s(15).replaceAll("\"", "").toInt,
        s(16).replaceAll("\"", "").toInt,
        s(17).replaceAll("\"", "").toInt,
        s(18).replaceAll("\"", "").toInt
    )
)

val sqlContext = new SQLContext(sc)
import sqlContext.implicits._

// convert to DataFrame and create temporal table
Set2Train.toDF().registerTempTable("Set2Train")
Set2Test.toDF().registerTempTable("Set2Test")
```

```
Set1Train.toDF().registerTempTable("Set1Train")
Set1Test.toDF().registerTempTable("Set1Test")
```

```
val output1 = sqlContext.sql(""" select
amount,hour1,state1,zip1,field1,domain_id,field2,hour2,flag1,total,field3,field4,field5,indicator1,indicator2,flag2,flag3,flag4,flag5 from
Set1Train
            INTERSECT
            select
amount,hour1,state1,zip1,field1,domain_id,field2,hour2,flag1,total,field3,field4,field5,indicator1,indicator2,flag2,flag3,flag4,flag5
            from Set1Test """)
```

```
output1.registerTempTable("people2")
sqlContext.cacheTable("people2")
sqlContext.sql("SELECT count(*) From people2").show

scala> sqlContext.sql("SELECT * From people2").show
+------+-----+------+----+------+--------------------+------+-----+-----+-----+------+------+------+----------+----------+-----+-----+-----+-----+
|amount|hour1|state1|zip1|field1|
domain_id|field2|hour2|flag1|total|field3|field4|field5|indicator1|indicator2|flag2|flag3|flag4|flag5|
+------+-----+------+----+------+--------------------+------+-----+-----+-----+------+------+------+----------+----------+-----+-----+-----+-----+
| 38.85|   20|    CT|  68|     3|         MADSOUL.COM|     1|   20|    1|38.85|  1599|     7|     0|         0|         0|    1|    1|    0|    1|
| 12.95|   21|    OR| 972|     3|        XVSQJIYR.COM|     0|   21|    1|12.95| -1440|     9|     0|         0|         0|    1|    0|    0|    1|
| 12.95|   15|    FL| 330|     3|     DADESCHOOLS.NET|     1|   15|    0|12.95|  5625|    26|     0|         0|         0|    1|    1|    0|    1|
| 38.85|   21|    DC| 200|     3|   ROGERLEEGROUP.COM|     0|   21|    1|38.85| -4099|     8|     0|         0|         0|    1|    0|    0|    1|
| 12.95|    9|    GA| 300|     3|REBDSUBMCWQXCPOB.COM|     1|    9|    1|12.95| -2392|     9|     9|         0|         0|    1|    1|    0|    1|
| 10.36|    0|    CA| 926|     3|           YAHOO.COM|     0|    0|    1|10.36|  4212|     7|     0|         1|         0|    1|    0|    0|    2|
+------+-----+------+----+------+--------------------+------+-----+-----+-----+------+------+------+----------+----------+-----+-----+-----+-----+
```

```
sqlContext.sql("""select state1,zip1,field1,domain_id from Set1Train
            INTERSECT
            select state1,zip1,field1,domain_id  from Set1Test limit 20""").show

+------+----+------+-----------+
|state1|zip1|field1|  domain_id|
+------+----+------+-----------+
|    FL| 333|     4|    AOL.COM|
|    VA| 201|     4|HOTMAIL.COM|
|    IL| 604|     4|  YAHOO.COM|
|    GA| 303|     4|    AOL.COM|
|    NY| 117|     4|    AOL.COM|
+------+----+------+-----------+
```

```
sqlContext.sql("""select
amount,hour1,state1,zip1,field1,email_id,field2,hour2,flag1,total,field3,field4,indicator1,indicator2,flag2,flag3,flag4,flag5 from
Set2Train
            INTERSECT
            select
amount,hour1,state1,zip1,field1,email_id,field2,hour2,flag1,total,field3,field4,indicator1,indicator2,flag2,flag3,flag4,flag5
```

```
                from Set2Test """).show

+------+-----+------+----+------+--------+------+-----+-----+-----+------+------+----------+----------+-----+-----+-----+-----+
|amount|hour1|state1|zip1|field1|email_id|field2|hour2|flag1|total|field3|field4|indicator1|indicator2|flag2|flag3|flag4|flag5|
+------+-----+------+----+------+--------+------+-----+-----+-----+------+------+----------+----------+-----+-----+-----+-----+
+------+-----+------+----+------+--------+------+-----+-----+-----+------+------+----------+----------+-----+-----+-----+-----+

sqlContext.sql("""select trim(customer_id) as c1 from Set2Train where email_id = 'zwzihwgzxohnq@cbbtr.com'
                INTERSECT
                select trim(customer_id) as c1 from Set2Test where email_id = 'zwzihwgzxohnq@cbbtr.com' limit 20""").show

+---+
| c1|
+---+
+---+


sqlContext.sql("""select substr(upper(trim(email_id)),1,10) as c1 from Set2Train
                INTERSECT
                select substr(upper(trim(email_id)),1,10) as c1 from Set2Test limit 20""").show

+----------+
|        c1|
+----------+
|EYLECFOESQ|
|IJOYZXJEPP|
|REDJHMBDZM|
|ZOTFCHXIQQ|
+----------+


sqlContext.sql("""select upper(trim(email_id)) as c1 from Set2Train  limit 20""").show

+--------------------+
|                  c1|
+--------------------+
|LUHXSODZMJHNG7@CO...|
|PFIXYIQFPVKCG@ZJY...|
|SHBJOLDCISWWM@AOL...|
|IPBVTDFKHHFWS@SBC...|
|IVFUVXIENNHDP@BEL...|
|GMLVCQEWYYCZT50@H...|
|CURBZPHDMPNYW@GMA...|
|PJATFYHRVHENR@YAH...|
|LILLTBVLOLCGA@ZPS...|
|OTXHMCLAIEMUB7@EA...|
|CURBZPHDMPNYW@GMA...|
|GBTJSCPUIPHLY@HOT...|
|QAQOQUCJIUNHR777@...|
|EPBRTOFQRHVCW1@TI...|
|BAIGKBZHIKMOY@COM...|
|UEQWHIACKHSFZ@HOT...|
|OKMOQHQLCMNKF@ALO...|
|ZVZHTWDTMVVAN14@H...|
|PPVDRTEDEXEPR@YAH...|
|IVFKCCNFBCORO1@YA...|
+--------------------+
sqlContext.sql("""select * from Set2Test where email_id = 'zwzihwgzxohnq@cbbtr.com'""").show

+------+-----+------+----+----------------+------+--------------------+------+-----+-----+-----+------+------+----------+----------+-----+-----+-----+-----+
|amount|hour1|state1|zip1|     customer_id|field1|
email_id|field2|hour2|flag1|total|field3|field4|indicator1|indicator2|flag2|flag3|flag4|flag5|
+------+-----+------+----+----------------+------+--------------------+------+-----+-----+-----+------+------+----------+----------+-----+-----+-----+-----+
|   0.0|    8|    LA| 707|1234567890197263|     4|zwzihwgzxohnq@cbb...|     0|    8|    0|  0.0| -5574|    11|         0|         0|    1|    1|    0|
3278|
|   0.0|   10|    LA| 707|1234567890197317|     4|zwzihwgzxohnq@cbb...|     0|   10|    0|  0.0|  1819|     9|         0|         0|    1|    1|    0|
3278|
|   0.0|   10|    LA| 708|1234567890197352|     4|zwzihwgzxohnq@cbb...|     0|   10|    1|  0.0|  3401|    15|         1|         0|    1|    1|    0|
3278|
|   0.0|   12|    LA| 705|1234567890197435|     4|zwzihwgzxohnq@cbb...|     1|   12|    0|  0.0|  4446|    13|         0|         0|    1|    1|    0|
3278|
|   0.0|   15|    LA| 708|1234567890197573|     4|zwzihwgzxohnq@cbb...|     0|   15|    0|  0.0|  3420|    12|         0|         0|    1|    1|    0|
3278|
|   0.0|   15|    LA| 708|1234567890197587|     4|zwzihwgzxohnq@cbb...|     1|   15|    0|  0.0|  1613|    12|         0|         0|    1|    1|    0|
3278|
|   0.0|   16|    LA| 708|1234567890197631|     4|zwzihwgzxohnq@cbb...|     0|   16|    0|  0.0| -3988|    12|         0|         0|    1|    1|    0|
3278|
|   0.0|    8|    LA| 708|1234567890197970|     4|zwzihwgzxohnq@cbb...|     0|    8|    0|  0.0|  2973|    11|         0|         0|    1|    1|    0|
3278|
|   0.0|    9|    MS| 395|1234567890197998|     4|zwzihwgzxohnq@cbb...|     0|    9|    1|  0.0| -6382|    10|         0|         0|    1|    1|    0|
3278|
|   0.0|   10|    LA| 708|1234567890198039|     4|zwzihwgzxohnq@cbb...|     1|   10|    0|  0.0|  -601|     8|         0|         0|    1|    1|    0|
3278|
|   0.0|   11|    LA| 708|1234567890198104|     4|zwzihwgzxohnq@cbb...|     0|   11|    0|  0.0| -4327|    12|         0|         0|    1|    1|    0|
3278|
|   0.0|   12|    LA| 703|1234567890198118|     4|zwzihwgzxohnq@cbb...|     1|   12|    0|  0.0| -4415|    10|         0|         1|    1|    1|    0|
3278|
|   0.0|   13|    LA| 708|1234567890198209|     4|zwzihwgzxohnq@cbb...|     1|   13|    0|  0.0|  2059|    13|         0|         0|    1|    1|    0|
3278|
|   0.0|   16|    LA| 707|1234567890198309|     4|zwzihwgzxohnq@cbb...|     1|   16|    0|  0.0|  1003|    16|         0|         1|    1|    1|    0|
3278|
|   0.0|    9|    LA| 704|1234567890198745|     4|zwzihwgzxohnq@cbb...|     0|    9|    0|  0.0| -1891|     9|         0|         0|    1|    1|    0|
3278|
|   0.0|   12|    LA| 707|1234567890198830|     4|zwzihwgzxohnq@cbb...|     0|   12|    0|  0.0|  2766|    13|         0|         0|    1|    1|    0|
3278|
|   0.0|   12|    LA| 708|1234567890198864|     4|zwzihwgzxohnq@cbb...|     0|   12|    0|  0.0|  1009|    10|         0|         0|    1|    1|    0|
3278|
|   0.0|   12|    LA| 708|1234567890198865|     4|zwzihwgzxohnq@cbb...|     0|   12|    0|  0.0|  1009|    10|         0|         0|    1|    1|    0|
3278|
|   0.0|   15|    LA| 708|1234567890199000|     4|zwzihwgzxohnq@cbb...|     1|   15|    1|  0.0|  1302|     6|         0|         0|    1|    1|    0|
3278|
|   0.0|   15|    LA| 708|1234567890199001|     4|zwzihwgzxohnq@cbb...|     1|   15|    1|  0.0|  1302|     6|         0|         0|    1|    1|    0|
3278|
+------+-----+------+----+----------------+------+--------------------+------+-----+-----+-----+------+------+----------+----------+-----+-----+-----+-----+
only showing top 20 rows

sqlContext.sql("""select distinct email_id as c1 from Set2Train where email_id = 'zwzihwgzxohnq@cbbtr.com'
            INTERSECT
            select distinct email_id as c1 from Set2Test where email_id = 'zwzihwgzxohnq@cbbtr.com' limit 20""").show

+--------------------+
|                  c1|
+--------------------+
|zwzihwgzxohnq@cbb...|
+--------------------+


val output2 = sqlContext.sql("""select distinct email_id as c1 from Set2Train
                  INTERSECT
                  select distinct email_id as c1 from Set2Test """)

output2.registerTempTable("people4")
sqlContext.cacheTable("people4")
scala> sqlContext.sql("SELECT * From people4").show
+--------------------+
|                  c1|
+--------------------+
|zwzihwgzxohnq@cbb...|
|ouekcnrdubsyz@hom...|
|zsxbringavcae@cbb...|
|dimqrwsvjwyql@ccc...|
|antihyknzxmva@cho...|
|hvaomlcrtkgws@cre...|
|pugrmyvovmvyp@cbb...|
|qcosrkuucvhwq@cbb...|
+--------------------+



val output3 = sqlContext.sql("""select distinct customer_id as c1 from Set2Train
                  INTERSECT
                  select distinct customer_id as c1 from Set2Test """)

output3.registerTempTable("people3")
sqlContext.cacheTable("people3")
sqlContext.sql("SELECT count(*) From people3").show

scala> sqlContext.sql("SELECT * From people3").show
+----------------+
|              c1|
+----------------+
|1234567890127208|
+----------------+

scala> val output4 = sqlContext.sql("SELECT customer_id , count(1) From Set2Train Group by customer_id having count(1) = 1")
output4: org.apache.spark.sql.DataFrame = [customer_id: string, _c1: bigint]

scala> output4.registerTempTable("people4")

scala> sqlContext.cacheTable("people4")

scala> sqlContext.sql("SELECT count(*) From people4").show
+-----+
|  _c0|
+-----+
|59082|
+-----+
scala> val output5 = sqlContext.sql("SELECT customer_id , count(1) From Set2Train Group by customer_id having count(1) >1")
output4: org.apache.spark.sql.DataFrame = [customer_id: string, _c1: bigint]

scala> output5.registerTempTable("people5")

scala> sqlContext.cacheTable("people5")

scala> sqlContext.sql("SELECT count(*) From people5").show
+-----+
|  _c0|
+-----+
|14647|
+-----+
scala> val output6 = sqlContext.sql("SELECT customer_id , count(1) From Set1Train Group by customer_id having count(1) = 1")
output4: org.apache.spark.sql.DataFrame = [customer_id: string, _c1: bigint]

scala> output6.registerTempTable("people6")

scala> sqlContext.cacheTable("people6")

scala> sqlContext.sql("SELECT count(*) From people6").show
+-----+
|  _c0|
+-----+
|59082|
+-----+


scala> val output7 = sqlContext.sql("SELECT customer_id , count(1) From Set2Train Group by customer_id having count(1) >1")
output4: org.apache.spark.sql.DataFrame = [customer_id: string, _c1: bigint]

scala> output7.registerTempTable("people7")

scala> sqlContext.cacheTable("people7")

scala> sqlContext.sql("SELECT count(*) From people7").show
+-----+
|  _c0|
+-----+
|14647|
+-----+

scala> val output8 = sqlContext.sql("SELECT domain_id , count(1) From Set1Train Group by domain_id having count(1) = 1")
output5: org.apache.spark.sql.DataFrame = [domain_id: string, _c1: bigint]

scala> output8.registerTempTable("people8")

scala> sqlContext.cacheTable("people8")

scala> sqlContext.sql("SELECT count(*) From people8").show
+----+
| _c0|
+----+
|5972|
+----+
scala> val output9 = sqlContext.sql("SELECT domain_id , count(1) From Set1Train Group by domain_id having count(1) > 1")
output5: org.apache.spark.sql.DataFrame = [domain_id: string, _c1: bigint]

scala> output9.registerTempTable("people9")

scala> sqlContext.cacheTable("people9")

scala> sqlContext.sql("SELECT count(*) From people9").show
+----+
| _c0|
+----+
|3838|
+----+
```

Section B

```
MATCH (n)
WITH n LIMIT 10000
OPTIONAL MATCH (n)-[r]->()
DELETE n,r

CREATE CONSTRAINT ON (person:Person) ASSERT person.id IS UNIQUE;

CREATE INDEX ON :Country(name);


USING PERIODIC COMMIT 1000
LOAD CSV WITH HEADERS FROM "file:///Users/Ritik/Desktop/Fraud_Dataset/Task2Train.csv" AS line
MATCH (person:Person { id: toInt(csvLine.personId)}),(movie:Movie { id: toInt(csvLine.movieId)})
CREATE (person)-[:PLAYED { role: csvLine.role }]->(movie)


CREATE CONSTRAINT ON (e:emailid) ASSERT e.name IS UNIQUE;
CREATE INDEX ON :custAttr1(name);

USING PERIODIC COMMIT 5000
LOAD CSV WITH HEADERS FROM "file:///Users/Ritik/Desktop/Fraud_Dataset/Task2Train.csv"
  AS line

WITH line

MERGE (c:customerid {name:line.custAttr1 ,  hour: line.hour1 , amount: line.amount, class: line.Class})
MERGE (e:emailid {name:line.custAttr2})
CREATE (c)-[r1:HAS_EMAIL]->(e)
MERGE (s:state {name:line.state1})
CREATE (c)-[r2:BELONGS_TO]->(s)
MERGE (z:zip {name:line.zip1})
CREATE (c)-[r3:HAS_ZIP]->(z)
RETURN c,r1,e ,s,r2,z ,r3;


MATCH (c:customerid)-[r1:HAS_EMAIL]->(e:emailid)
WHERE c.class = '1'
```

```
RETURN c , r1 , e LIMIT 100


START n=node(*)
MATCH (n:customerid)-[r:HAS_EMAIL]->(x:emailid)
WITH n,r,x, count(x) as c
where n.class = '1'
RETURN n,r,x,c
ORDER BY c DESC

//Nodes with same properties
match (n:Label)
with n.prop as prop, collect(n) as nodelist, count(*) as count
where count > 1
return prop, nodelist, count;


MATCH (c:customerid)-[r:HAS_EMAIL]->()
WITH c.customerid as customerid, count(*) as idCount
WHERE idCount>1
RETURN customerid,idCount


MATCH           (accountHolder:customerid)-[]->(contactInformation:emailid)
WITH            contactInformation,
                          count(accountHolder) AS RingSize
MATCH           (contactInformation:emailid)<-[]-(accountHolder)
WITH            collect(accountHolder.name) AS AccountHolders,
                          contactInformation, RingSize
WHERE           RingSize > 1
RETURN          AccountHolders AS FraudRing,
                          labels(contactInformation) AS ContactType,
                          RingSize
ORDER BY        RingSize DESC limit 1000

MATCH           (accountHolder:customerid)-[]->(contactInformation:emailid)
WITH            contactInformation,
                          count(distinct accountHolder) AS RingSize
MATCH           (contactInformation:emailid)<-[]-(accountHolder)
WITH            collect(distinct accountHolder.name) AS AccountHolders,
                          contactInformation, RingSize,
                          SUM(accountHolder.amount) as FinancialRisk
WHERE           RingSize > 1
RETURN          AccountHolders AS FraudRing,
                          labels(contactInformation) AS ContactType,
                          RingSize , FinancialRisk
ORDER BY        RingSize DESC limit 1000


MATCH           (accountHolder:customerid)-[]->(contactInformation:emailid)
WITH            contactInformation,
                          count(distinct accountHolder) AS RingSize
MATCH           (contactInformation:emailid)<-[]-(accountHolder)
WITH            collect(distinct accountHolder.name) AS AccountHolders,
                          contactInformation, RingSize,
                          SUM(toint(accountHolder.amount)) as FinancialRisk
WHERE           RingSize > 1
```

```
RETURN          AccountHolders AS FraudRing,
                        labels(contactInformation) AS ContactType,
                        RingSize , FinancialRisk
ORDER BY        RingSize DESC limit 1000


MATCH           (accountHolder:customerid)-[]->(contactInformation:emailid)
WITH            contactInformation,
                        count(distinct accountHolder) AS RingSize
MATCH           (contactInformation:emailid)<-[]-(accountHolder)
WITH            collect(distinct accountHolder.name) AS AccountHolders,
                        contactInformation, RingSize,
                        SUM(toint(accountHolder.amount)) as FinancialRisk
WHERE           RingSize > 1
RETURN          AccountHolders AS FraudRing,
                        contactInformation.name AS ContactType,
                        RingSize , FinancialRisk
ORDER BY        FinancialRisk DESC limit 1000
```

Section C

```
from py2neo import authenticate, Graph


authenticate("localhost:7474", "neo4j", "as8172")


sgraph = Graph("http://localhost:7474/db/data/")


query = """
MATCH (c:customerid)-[r1:HAS_EMAIL]->(e:emailid)
WHERE c.class = '1'
RETURN c.name as id , r1 , e LIMIT 100
"""

data = sgraph.cypher.execute(query)

get_ipython().magic(u'load_ext cypher')


get_ipython().run_cell_magic(u'cypher', u' http://neo4j:as8172@localhost:7474/db/data', u"MATCH
(c:customerid)-[r1:HAS_EMAIL]->(e:emailid)\nWHERE c.class = '1'\nRETURN c.name as id , r1 , e LIMIT 100")


results = get_ipython().magic(u'cypher http://neo4j:as8172@localhost:7474/db/data MATCH
(c:customerid)-[r1:HAS_EMAIL]->(e:emailid)             RETURN c.name as id , r1 , e LIMIT 100')

df = results.get_dataframe()


import networkx as nx
```

```
get_ipython().magic(u'matplotlib inline')

results = get_ipython().magic(u'cypher http://neo4j:as8172@localhost:7474/db/data MATCH p =
(c:customerid)-[r1:HAS_EMAIL]->(e:emailid) RETURN p  limit 10')

g = results.get_graph()

nx.draw(g)


# In[15]:

g.nodes(data=True)


# In[16]:

from py2neo import Graph as PGraph
from igraph import Graph as IGraph

neo4j = PGraph()

query = """
MATCH (c:customerid)-[r1:HAS_EMAIL]->(e:emailid)
WHERE c.class = '1'
RETURN c.name as id , r1 , e LIMIT 100
"""

data = neo4j.cypher.execute(query)
data


# In[17]:

ig = IGraph.TupleList(data)
ig


# In[18]:

best = ig.vs.select(_degree = ig.maxdegree())["name"]
best


# In[19]:

import jgraph

data = sgraph.cypher.execute("MATCH (c:customerid)-[r1:HAS_EMAIL]->(e:emailid) WHERE c.class = '1' RETURN c.name as id ,
e.name as email LIMIT 100")
data = [tuple(x) for x in data]

jgraph.draw(data)
```

Section D

```
library(dataiku)
library(rpart)
library(lattice)
library(ggplot2)
library(caret)
library(e1071)
library(DMwR)

# Recipe inputs
data <- dkuReadDataset("Task2Train", samplingMethod="head", nbRows=100000)


intrain<-createDataPartition(y = data$Class,p = .70,list = FALSE,
                             times = 1)
trainSplit<-data[intrain,]
testSplit<-data[-intrain,]

table(trainSplit$Class)
table(testSplit$Class)

# Recipe outputs
dkuWriteDataset(trainSplit,"Training")
dkuWriteDataset(testSplit,"Testing")
```

Section E

```
library(dataiku)

#install.packages("ROSE", repos="http://cran.rstudio.com/")

library(ROSE)
library(rpart)
library(caret)
library(lattice)
library(ggplot2)

# Recipe inputs
#training <- dkuReadDataset("Training", samplingMethod="head", nbRows=100000)

path <- "/home/dataiku/dss/uploads/FRAUDANALYSISUSINGR/datasets"
setwd(path)
getwd()

data <- read.csv('Task2Train/Task2Train.csv')


intrain<-createDataPartition(y=data$Class,p=0.7,list=FALSE)
training<-data[intrain,]
testing<-data[-intrain,]
```

```
data_balanced_under <- ovun.sample(Class ~ ., data = training, method = "under", N = 3708, seed = 1)$data
table(data_balanced_under$Class)
data_balanced_over <- ovun.sample(Class ~ ., data = training, method = "over",N = 136292)$data
table(data_balanced_over$Class)
data_balanced_both <- ovun.sample(Class ~ ., data = training, method = "both", p=0.5, N=5000, seed = 1)$data
table(data_balanced_both$Class)
data.rose <- ROSE(Class ~ ., data = training, seed = 1)$data
table(data.rose$Class)

# Recipe outputs
dkuWriteDataset(data_balanced_under,"data_balnace_under")
dkuWriteDataset(data_balanced_over,"data_balance_over")
dkuWriteDataset(data_balanced_both,"data_balance_both1")
dkuWriteDataset(data.rose,"data_balance_rose")
```

Section F

```
library(dataiku)
library(ROSE)
library(rpart)
library(caret)
library(lattice)
library(ggplot2)

# Recipe inputs
data_rose <- dkuReadDataset("data_balance_rose", samplingMethod="head", nbRows=100000)

tree.rose <- rpart(Class ~ ., data = data_rose)

# Recipe outputs
model_rose <- dkuManagedFolderPath("ldmANycK")
setwd(model_rose)
system("rm -rf *")
path <- paste(model_rose, 'model.RData', sep="/")
save(tree.rose, file = path)
```

Section G

```
### ROSE
pred.tree.rose <- predict(tree.rose, newdata = test.ml)
pred_tree_rose <- as.data.frame(cbind(testing , round(pred.tree.rose)))
pred_tree_rose <- rename ( pred_tree_rose , c("round(pred.tree.rose)"="predicted"))

# Recipe outputs
dkuWriteDataset(pred_tree_rose,"test_rose_score")

#### Over
```

```
tree.over <- rpart(Class ~ ., data = data_balanced_over)
pred.tree.over <- predict(tree.over, newdata = test.ml)
pred_tree_over <- as.data.frame(cbind(testing , round(pred.tree.over)))
pred_tree_over <- rename ( pred_tree_over , c("round(pred.tree.over)"="predicted"))

# Recipe outputs
dkuWriteDataset(pred_tree_over,"test_over_score")


#### Under

tree.under <- rpart(Class ~ ., data = data_balanced_under)
pred.tree.under <- predict(tree.under, newdata = test.ml)
pred_tree_under <- as.data.frame(cbind(testing , round(pred.tree.under)))
pred_tree_under <- rename ( pred_tree_under , c("round(pred.tree.under)"="predicted"))

# Recipe outputs
dkuWriteDataset(pred_tree_under,"test_under_score")

#### Hybrid

tree.both <- rpart(Class ~ ., data = data_balanced_both)
pred.tree.both <- predict(tree.both, newdata = test.ml)
pred_tree_both <- as.data.frame(cbind(testing , round(pred.tree.both)))
pred_tree_both <- rename ( pred_tree_both , c("round(pred.tree.both)"="predicted"))

# Recipe outputs
dkuWriteDataset(pred_tree_both,"test_both_score")
```

Section H

```
library(SparkR)
library(dataiku)
library(dataiku.spark)
library(rpart)
library(lattice)
library(ggplot2)
library(caret)
library(e1071)
library(DMwR)

sc <- sparkR.init()
sqlContext <- sparkRSQL.init(sc)


path <- "/home/dataiku/dss/uploads/FRAUDANALYSISUSINGR/datasets"
setwd(path)
getwd()

data <- read.csv('Task2Train/Task2Train.csv')
head(data)

intrain<-createDataPartition(y = data$Class,p = .70,list = FALSE,
                   times = 1)
```

```
trainSplit<-data[intrain,]
testSplit<-data[-intrain,]


trainSplit$Class <- as.factor(trainSplit$Class)
trainSplit <- SMOTE(Class ~ ., trainSplit, perc.over = 100, perc.under=200)
trainSplit$Class <- as.numeric(trainSplit$Class)

df1 <- createDataFrame(sqlContext, trainSplit)

# Recipe outputs
dkuSparkWriteDataset(df1,"TrainSplitSmot")

Data.smote <- TrainSplitSmot
tree.smote <- rpart(Class ~ ., data = data.smote)

pred.tree.smote <- predict(tree.smote, newdata = testSplit)


pred_tree_smote <- as.data.frame(cbind(testSplit , round(pred.tree.smote)))

pred_tree_smote <- rename ( pred_tree_smote , c("round(pred.tree.smote)"="predicted"))

# Recipe outputs
dkuWriteDataset(pred_tree_smote,"test_smote_score")
```

Section I

```
library(dataiku)
#install.packages("h2o", repos="http://cran.rstudio.com/")
library(h2o)
library(rpart)
library(lattice)
library(ggplot2)
library(caret)
library(e1071)
library(DMwR)

# Recipe inputs
test <- dkuReadDataset("Testing", samplingMethod="head", nbRows=100000)
train <- dkuReadDataset("Training", samplingMethod="head", nbRows=100000)



#intrain<-createDataPartition(y = df$Class,p = .70,list = FALSE,
           #     times = 1)
#train<-df[intrain,]
#test<-df[-intrain,]


#-----------------------------------------------------------------
# Settings
#-----------------------------------------------------------------
target.variable <- 'Class'
```

```
features.num <- c(
   'amount', 'field1', 'field2', 'field3', 'field4', 'flag1',
   'flag2', 'flag3', 'flag4', 'flag5', 'indicator1', 'indicator2'
)

features.cat <- c(
   'state1', 'zip1', 'hour1', 'custAttr1' , 'custAttr2'
)

#-----------------------------------------------------------------
# Preprocessing
#-----------------------------------------------------------------
train[features.cat]    <- lapply(train[features.cat], as.factor)
train[features.num]    <- lapply(train[features.num], as.double)
train[target.variable] <- lapply(train[target.variable], as.factor)
train.ml <- train[c(features.cat, features.num, target.variable)]


test[features.cat]    <- lapply(test[features.cat], as.factor)
test[features.num]    <- lapply(test[features.num], as.double)
test[target.variable] <- lapply(test[target.variable], as.factor)
test.ml <- test[c(features.cat, features.num)]

#------------------------------------------------------------------------
# TRAINING
#------------------------------------------------------------------------
localH2O <- h2o.init(nthreads = -1)

as.h2o(train.ml, destination_frame = 'h2otrain')

h2o.model <- h2o.gbm(
   x = c(features.cat, features.num),
   y = target.variable,
   "h2otrain",
   distribution = "bernoulli",
   tweedie_power = 1.5,
   ntrees = 50,
   max_depth = 5,
   min_rows = 10,
   learn_rate = 0.1,
   sample_rate = 1,
   col_sample_rate = 1,
   nbins = 20,
   nbins_cats = 1024
)

print(h2o.model)


#-----------------------------------------------------------------
# Predictions
#-----------------------------------------------------------------
h2o.test <- as.h2o(test.ml, destination_frame = 'h2otest')
preds <- h2o.predict(h2o.model, h2o.test)
predictions <- as.data.frame(preds)
results <- cbind(test, predictions)
```

```
# Recipe outputs
dkuWriteDataset(results,"Test_GBM_Score")
```

Section J

```
library(dataiku)
#install.packages("devtools", repos="http://cran.rstudio.com/")
library(devtools)
#install_github("h2oai/h2o-3/h2o-r/ensemble/h2oEnsemble-package")
#install.packages("h2o", repos="http://cran.rstudio.com/")
devtools::install_github("ledell/cvAUC")
library(cvAUC)
library(h2oEnsemble)
library(h2o)
library(rpart)
library(lattice)
library(ggplot2)
library(caret)
library(e1071)
library(DMwR)

# Recipe inputs
train <- dkuReadDataset("Training", samplingMethod="head", nbRows=100000)
test <- dkuReadDataset("Testing", samplingMethod="head", nbRows=100000)




#------------------------------------------------------------------
# Settings
#------------------------------------------------------------------
target.variable <- 'Class'

features.num <- c(
   'amount', 'field1', 'field2', 'field3', 'field4', 'flag1',
   'flag2', 'flag3', 'flag4', 'flag5', 'indicator1', 'indicator2'
)

features.cat <- c(
   'state1', 'zip1', 'hour1', 'custAttr1' , 'custAttr2'
)


#------------------------------------------------------------------
# Preprocessing
#------------------------------------------------------------------

train[target.variable] <- lapply(train[target.variable], as.factor)
train.ml <- train[c(features.cat, features.num, target.variable)]
```

```
test.ml <- test[c(features.cat, features.num)]

#---------------------------------------------------------------------------
# TRAINING
#---------------------------------------------------------------------------
localH2O <- h2o.init(nthreads = -1)
h2o.removeAll()

learner <- c("h2o.glm.wrapper", "h2o.randomForest.wrapper",
             "h2o.gbm.wrapper", "h2o.deeplearning.wrapper")
metalearner <- "h2o.glm.wrapper"

as.h2o(train.ml, destination_frame = 'h2otrain')

fit <- h2o.ensemble(x = 1:17,
                        y = 18,
                 "h2otrain" ,
                 family = "binomial",
                 learner = learner,
                 metalearner = metalearner,
                 cvControl = list(V = 5))


#------------------------------------------------------------------
# Predictions
#------------------------------------------------------------------
h2o.test <- as.h2o(test.ml, destination_frame = 'h2otest')


pred <- predict(fit, h2o.test)
head(pred)


predicted <- as.data.frame(pred$pred)[,1]  #third column, p1 is P(Y==1)
head(predicted)

results <- cbind(test, predicted)

labels <- as.data.frame(test[,20])[,1]
#results <- cbind(as.data.frame(h2o.test), as.data.frame(pred))

predictions <- as.data.frame(pred$pred)[,3]  #third column, p1 is P(Y==1)

# Ensemble test AUC
cvAUC::AUC(predictions = predictions , labels = labels)

L <- length(learner)
auc <- sapply(seq(L), function(l) cvAUC::AUC(predictions = as.data.frame(pred$basepred)[,l], labels = labels))
data.frame(learner, auc)

# Recipe outputs
dkuWriteDataset(results,"Test_Ensemble_Score")
h2o.shutdown()
```

Section K

```
library(dataiku)
library(h2o)
library(rpart)
library(lattice)
library(ggplot2)
library(caret)
library(e1071)
library(DMwR)

# Recipe inputs
test <- dkuReadDataset("Testing", samplingMethod="head", nbRows=100000)
train <- dkuReadDataset("Training", samplingMethod="head", nbRows=100000)

#-------------------------------------------------------------------
# Settings
#-------------------------------------------------------------------
target.variable <- 'Class'

features.num <- c(
    'amount', 'field1', 'field2', 'field3', 'field4', 'flag1',
    'flag2', 'flag3', 'flag4', 'flag5', 'indicator1', 'indicator2'
)

features.cat <- c(
    'state1', 'zip1', 'hour1', 'custAttr1' , 'custAttr2'
)


#-------------------------------------------------------------------
# Preprocessing
#-------------------------------------------------------------------
train[target.variable] <- lapply(train[target.variable], as.factor)

test.ml <- test[c(features.cat, features.num)]

#--------------------------------------------------------------------------
# TRAINING
#--------------------------------------------------------------------------
localH2O <- h2o.init(nthreads = -1, max_mem_size="5G")

h2o.removeAll()

as.h2o(train, destination_frame = 'h2otrain')

h2o.model <- h2o.deeplearning(x = 1:19,
                        y = 20,
                        "h2otrain",
                         balance_classes = TRUE,
                        activation = 'Rectifier',
                       hidden = c(50,50,50,50,50,50),
                       epochs = 500)
```

```
print(h2o.model)

#------------------------------------------------------------------
# Predictions
#------------------------------------------------------------------
h2o.test <- as.h2o(test.ml, destination_frame = 'h2otest')
preds <- h2o.predict(h2o.model, h2o.test)
predictions <- as.data.frame(preds)
results <- cbind(test, predictions)

# Recipe outputs
dkuWriteDataset(results,"Test_Deep_learning")
h2o.shutdown()
```

Section L

```
library(dataiku)
library(h2o)
library(rpart)
library(lattice)
library(ggplot2)
library(caret)
library(e1071)
library(DMwR)

# Recipe inputs
test <- dkuReadDataset("Testing", samplingMethod="head", nbRows=100000)
train <- dkuReadDataset("Training", samplingMethod="head", nbRows=100000)

head(train)


#------------------------------------------------------------------
# Settings
#------------------------------------------------------------------
target.variable <- 'Class'

features.num <- c(
   'amount', 'field1', 'field2', 'field3', 'field4', 'flag1',
   'flag2', 'flag3', 'flag4', 'flag5', 'indicator1', 'indicator2'
)


# TRAINING
#--------------------------------------------------------------------------
localH2O <- h2o.init(nthreads = -1, max_mem_size="5G")

h2o.removeAll()

as.h2o(train.ml, destination_frame = 'h2otrain')
features.cat <- c(
   'state1', 'zip1', 'hour1', 'custAttr1' , 'custAttr2'
)


#-------------------------------------------------------------------
# Preprocessing
#-------------------------------------------------------------------
train[target.variable] <- lapply(train[target.variable], as.factor)
train.ml <- train[c(features.cat, features.num, target.variable)]
test.ml <- test[c(features.cat, features.num)]

#--------------------------------------------------------------------------

h2o.model <- h2o.deeplearning(
                                        x=1:17,
                                        y=18,
                                        training_frame="h2otrain",
                                        hidden=c(50,50,50,50,50,50),
                                        epochs=500,
                                        nfolds=5,
                                        balance_classes = TRUE,
                                        fold_assignment="Stratified" # can be "AUTO", "Modulo", "Random" or "Stratified"
                                        )

print(h2o.model)


#-------------------------------------------------------------------
# Predictions
#-------------------------------------------------------------------
h2o.test <- as.h2o(test.ml, destination_frame = 'h2otest')
preds <- h2o.predict(h2o.model, h2o.test)
predictions <- as.data.frame(preds)
results <- cbind(test, predictions)



# Recipe outputs
dkuWriteDataset(results,"Deep_Learning_Cross_Validation")
h2o.shutdown()
```

Section M

```
library(dataiku)
library(h2o)
library(rpart)
library(lattice)
library(ggplot2)
library(caret)
library(e1071)
library(DMwR)

# Recipe inputs
```

```
test <- dkuReadDataset("Testing", samplingMethod="head", nbRows=100000)
train <- dkuReadDataset("Training", samplingMethod="head", nbRows=100000)

head(train)


#-----------------------------------------------------------------
# Settings
#-----------------------------------------------------------------
target.variable <- 'Class'

features.num <- c(
    'amount', 'field1', 'field2', 'field3', 'field4', 'flag1',
    'flag2', 'flag3', 'flag4', 'flag5', 'indicator1', 'indicator2'
)

features.cat <- c(
    'state1', 'zip1', 'hour1', 'custAttr1' , 'custAttr2'
)


#-----------------------------------------------------------------
# Preprocessing
#-----------------------------------------------------------------

train[target.variable] <- lapply(train[target.variable], as.factor)
train.ml <- train[c(features.cat, features.num, target.variable)]
test.ml <- test[c(features.cat, features.num)]

#------------------------------------------------------------------------
# TRAINING
#------------------------------------------------------------------------
localH2O <- h2o.init(nthreads = -1, max_mem_size="5G")

h2o.removeAll()

as.h2o(train.ml, destination_frame = 'h2otrain')

hyper_params <- list(
  activation=c("Rectifier","Tanh","Maxout","RectifierWithDropout","TanhWithDropout","MaxoutWithDropout"),
  hidden=list(c(20,20),c(50,50),c(30,30,30),c(25,25,25,25),c(50,50,50,50,50,50)),
  epochs=500,
  input_dropout_ratio=c(0,0.05),
  l1=seq(0,1e-4,1e-6),
  l2=seq(0,1e-4,1e-6)
)




search_criteria = list(strategy = "RandomDiscrete", max_runtime_secs = 360, max_models = 100, seed=1234, stopping_rounds=5,
stopping_tolerance=1e-2)

dl_random_grid <- h2o.grid(
                              algorithm="deeplearning",
                              grid_id = "dl_grid_random",
                              training_frame= as.h2o(train.ml) ,
                              x=1:17,
                              y=18,
                              #epochs=500,
                              balance_classes = TRUE,
                              stopping_metric="logloss",
                              stopping_tolerance=1e-2,        ## stop when logloss does not improve by >=1% for 2 scoring events
                              stopping_rounds=2,
                              #score_validation_samples=10000, ## downsample validation set for faster scoring
                              score_duty_cycle=0.025,         ## don't score more than 2.5% of the wall time
                              max_w2=10,                      ## can help improve stability for Rectifier
                              hyper_params = hyper_params,
                              search_criteria = search_criteria
                            )

grid <- h2o.getGrid("dl_grid_random",sort_by="logloss",decreasing=FALSE)
grid

grid@summary_table[1,]
best_model <- h2o.getModel(grid@model_ids[[1]]) ## model with lowest logloss
best_model

print(best_model@allparameters)
print(h2o.performance(best_model, valid=T))
print(h2o.logloss(best_model, valid=T))


h2o.confusionMatrix(best_model,valid=T)
best_params <- best_model@allparameters
best_params$activation
best_params$hidden
best_params$input_dropout_ratio
best_params$l1
best_params$l2


#-------------------------------------------------------------------
# Predictions
#-------------------------------------------------------------------
h2o.test <- as.h2o(test.ml, destination_frame = 'h2otest')
preds <- h2o.predict(best_model, h2o.test)
predictions <- as.data.frame(preds)
results <- cbind(test, predictions)



# Recipe outputs
dkuWriteDataset(results,"Deep_Learning_Grid")
h2o.shutdown()
```

Section N

```
# -*- coding: utf-8 -*-
import dataiku
import pandas as pd, numpy as np
```

```
from dataiku import pandasutils as pdu


# -*- coding: utf-8 -*-
import dataiku
import numpy as np
import pandas as pd
from sklearn.metrics import roc_auc_score
from sklearn.metrics import recall_score
from sklearn.metrics import matthews_corrcoef
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix

# Recipe inputs
df1 = dataiku.Dataset("test_smote_score").get_dataframe()
df2 = dataiku.Dataset("test_both_score").get_dataframe()
df3 = dataiku.Dataset("test_rose_score").get_dataframe()
df4 = dataiku.Dataset("test_over_score").get_dataframe()
df5 = dataiku.Dataset("test_under_score").get_dataframe()
df6 = dataiku.Dataset("Test_RF_score").get_dataframe()
df7 = dataiku.Dataset("Test_GBM_Score").get_dataframe()
df8 = dataiku.Dataset("Test_Deep_learning").get_dataframe()
df9 = dataiku.Dataset("Test_Ensemble_Score").get_dataframe()
df10 = dataiku.Dataset("H2O_RF").get_dataframe()
df11 = dataiku.Dataset("H2O_GLM").get_dataframe()
df12 = dataiku.Dataset("Deep_Learning_Cross_Validation").get_dataframe()
df13 = dataiku.Dataset("Deep_Learning_Grid").get_dataframe()


# Measure AUC
print '[+] Building AUC for model 1...'
auc1 = roc_auc_score(df1['Class'].astype(bool).values, df1['predicted'].values)
auc1 = pd.Series(auc1, name='auc')
#auc11 = pd.Series('Sample Smote', name='Model')
auc1 = pd.DataFrame(auc1)
#auc1 = [ auc11,auc1]
auc1['model'] = 'Sample Smote'


print confusion_matrix(df1['Class'].astype(bool).values, df1['predicted'].values)
rec1 = recall_score(df1['Class'].astype(bool).values, df1['predicted'].values)
rec1 = pd.Series(rec1, name='rec')
auc1[ 'Recal'] = pd.DataFrame(rec1)


mcc1 = matthews_corrcoef(df1['Class'].astype(bool).values, df1['predicted'].values)
mcc1 = pd.Series(mcc1, name='mcc')
auc1[ 'MCC'] = pd.DataFrame(mcc1)

pc1 = precision_score(df1['Class'].astype(bool).values, df1['predicted'].values)
pc1 = pd.Series(pc1, name='pc1')
auc1[ 'Precision'] = pd.DataFrame(pc1)

f1_1 = f1_score(df1['Class'].astype(bool).values, df1['predicted'].values)
f1_1 = pd.Series(f1_1, name='f1')
auc1[ 'F Score'] = pd.DataFrame(f1_1)
```

```
print '[+] Building AUC for model 2...'
auc2 = roc_auc_score(df2['Class'].astype(bool).values, df2['predicted'].values)
auc2 = pd.Series(auc2, name='auc')
#auc21 = pd.Series('Sample both', name='Model')
auc2 = pd.DataFrame(auc2)
auc2['model'] = 'Sample both'
#auc2 = [ auc21,auc2]

print confusion_matrix(df2['Class'].astype(bool).values, df2['predicted'].values)
rec2 = recall_score(df2['Class'].astype(bool).values, df2['predicted'].values)
rec2 = pd.Series(rec2, name='rec')
auc2[ 'Recal'] = pd.DataFrame(rec2)

mcc2 = matthews_corrcoef(df2['Class'].astype(bool).values, df2['predicted'].values)
mcc2 = pd.Series(mcc2, name='mcc')
auc2[ 'MCC'] = pd.DataFrame(mcc2)

pc2 = precision_score(df2['Class'].astype(bool).values, df2['predicted'].values)
pc2 = pd.Series(pc2, name='pc')
auc2[ 'Precision'] = pd.DataFrame(pc2)

f1_2 = f1_score(df2['Class'].astype(bool).values, df2['predicted'].values)
f1_2 = pd.Series(f1_2, name='f1')
auc2[ 'F Score'] = pd.DataFrame(f1_2)

print '[+] Building AUC for model 3...'
auc3 = roc_auc_score(df3['Class'].astype(bool).values, df3['predicted'].values)
auc3 = pd.Series(auc3, name='auc')
#auc3 = pd.Series('Sample Rose', name='Model')
auc3 = pd.DataFrame(auc3)
auc3['model'] = 'Sample Rose'


print confusion_matrix(df3['Class'].astype(bool).values, df3['predicted'].values)
rec3 = recall_score(df3['Class'].astype(bool).values, df3['predicted'].values)
rec3 = pd.Series(rec3, name='rec')
auc3[ 'Recal'] = pd.DataFrame(rec3)


mcc3 = matthews_corrcoef(df3['Class'].astype(bool).values, df3['predicted'].values)
mcc3 = pd.Series(mcc3, name='mcc')
auc3[ 'MCC'] = pd.DataFrame(mcc3)

pc3 = precision_score(df3['Class'].astype(bool).values, df3['predicted'].values)
pc3 = pd.Series(pc3, name='pc')
auc3[ 'Precision'] = pd.DataFrame(pc3)

f1_3 = f1_score(df3['Class'].astype(bool).values, df3['predicted'].values)
f1_3 = pd.Series(f1_3, name='f1')
auc3[ 'F Score'] = pd.DataFrame(f1_3)

print '[+] Building AUC for model 4...'
auc4 = roc_auc_score(df4['Class'].astype(bool).values, df4['predicted'].values)
auc4 = pd.Series(auc4, name='auc')
#auc4 = pd.Series('Sample Over', name='Model')
auc4 = pd.DataFrame(auc4)
```

```
auc4['model'] = 'Sample Over'


print confusion_matrix(df4['Class'].astype(bool).values, df4['predicted'].values)
rec4 = recall_score(df4['Class'].astype(bool).values, df4['predicted'].values)
rec4 = pd.Series(rec4, name='rec')
auc4[ 'Recal'] = pd.DataFrame(rec4)


mcc4 = matthews_corrcoef(df4['Class'].astype(bool).values, df4['predicted'].values)
mcc4 = pd.Series(mcc4, name='mcc')
auc4[ 'MCC'] = pd.DataFrame(mcc4)

pc4 = precision_score(df4['Class'].astype(bool).values, df4['predicted'].values)
pc4 = pd.Series(pc4, name='pc')
auc4[ 'Precision'] = pd.DataFrame(pc4)

f1_4 = f1_score(df4['Class'].astype(bool).values, df4['predicted'].values)
f1_4 = pd.Series(f1_4, name='f1')
auc4[ 'F Score'] = pd.DataFrame(f1_4)


print '[+] Building AUC for model 5...'
auc5 = roc_auc_score(df5['Class'].astype(bool).values, df5['predicted'].values)
auc5 = pd.Series(auc5, name='auc')
#auc5 = pd.Series('Sample Under', name='Model')
auc5 = pd.DataFrame(auc5)
auc5['model'] = 'Sample Under'

print confusion_matrix(df5['Class'].astype(bool).values, df5['predicted'].values)
rec5 = recall_score(df5['Class'].astype(bool).values, df5['predicted'].values)
rec5 = pd.Series(rec5, name='rec')
auc5[ 'Recal'] = pd.DataFrame(rec5)


mcc5 = matthews_corrcoef(df5['Class'].astype(bool).values, df5['predicted'].values)
mcc5 = pd.Series(mcc5, name='mcc')
auc5[ 'MCC'] = pd.DataFrame(mcc5)

pc5 = precision_score(df5['Class'].astype(bool).values, df5['predicted'].values)
pc5 = pd.Series(pc5, name='pc')
auc5[ 'Precision'] = pd.DataFrame(pc5)

f1_5 = f1_score(df5['Class'].astype(bool).values, df5['predicted'].values)
f1_5 = pd.Series(f1_5, name='f1')
auc5[ 'F Score'] = pd.DataFrame(f1_5)

print '[+] Building AUC for model 6...'
auc6 = roc_auc_score(df6['Class'].astype(bool).values, df6['prediction'].values)
auc6 = pd.Series(auc6, name='auc')
#auc6 = pd.Series('RF', name='Model')
auc6 = pd.DataFrame(auc6)
auc6['model'] = 'DSS RF'


print confusion_matrix(df6['Class'].astype(bool).values, df6['prediction'].values)
rec6 = recall_score(df6['Class'].astype(bool).values, df6['prediction'].values)
```

```
rec6 = pd.Series(rec6, name='rec')
auc6[ 'Recal'] = pd.DataFrame(rec6)


mcc6 = matthews_corrcoef(df6['Class'].astype(bool).values, df6['prediction'].values)
mcc6 = pd.Series(mcc6, name='mcc')
auc6[ 'MCC'] = pd.DataFrame(mcc6)

pc6 = precision_score(df6['Class'].astype(bool).values, df6['prediction'].values)
pc6 = pd.Series(pc6, name='pc')
auc6[ 'Precision'] = pd.DataFrame(pc6)

f1_6 = f1_score(df6['Class'].astype(bool).values, df6['prediction'].values)
f1_6 = pd.Series(f1_6, name='f1')
auc6[ 'F Score'] = pd.DataFrame(f1_6)


print '[+] Building AUC for model 7...'
auc7 = roc_auc_score(df7['Class'].astype(bool).values, df7['predict'].values)
auc7 = pd.Series(auc7, name='auc')
#auc7 = pd.Series('H2O GBM', name='Model')
auc7 = pd.DataFrame(auc7)
auc7['model'] = 'H2O GBM'

print confusion_matrix(df7['Class'].astype(bool).values, df7['predict'].values)
rec7 = recall_score(df7['Class'].astype(bool).values, df7['predict'].values)
rec7 = pd.Series(rec7, name='rec')
auc7[ 'Recal'] = pd.DataFrame(rec7)


mcc7 = matthews_corrcoef(df7['Class'].astype(bool).values, df7['predict'].values)
mcc7 = pd.Series(mcc7, name='mcc')
auc7[ 'MCC'] = pd.DataFrame(mcc7)

pc7 = precision_score(df7['Class'].astype(bool).values, df7['predict'].values)
pc7 = pd.Series(pc7, name='pc')
auc7[ 'Precision'] = pd.DataFrame(pc7)

f1_7 = f1_score(df7['Class'].astype(bool).values, df7['predict'].values)
f1_7 = pd.Series(f1_7, name='f1')
auc7[ 'F Score'] = pd.DataFrame(f1_7)

print '[+] Building AUC for model 8...'
auc8 = roc_auc_score(df8['Class'].astype(bool).values, df8['predict'].values)
auc8 = pd.Series(auc8, name='auc')
#auc8 = pd.Series('Deep Learning', name='Model')
auc8 = pd.DataFrame(auc8)
auc8['model'] = 'H2O Deep Learning'


print confusion_matrix(df8['Class'].astype(bool).values, df8['predict'].values)
rec8 = recall_score(df8['Class'].astype(bool).values, df8['predict'].values)
rec8 = pd.Series(rec8, name='rec')
auc8[ 'Recal'] = pd.DataFrame(rec8)


mcc8 = matthews_corrcoef(df8['Class'].astype(bool).values, df8['predict'].values)
```

```
mcc8 = pd.Series(mcc8, name='mcc')
auc8[ 'MCC'] = pd.DataFrame(mcc8)

pc8 = precision_score(df8['Class'].astype(bool).values, df8['predict'].values)
pc8 = pd.Series(pc8, name='pc')
auc8[ 'Precision'] = pd.DataFrame(pc8)

f1_8 = f1_score(df8['Class'].astype(bool).values, df8['predict'].values)
f1_8 = pd.Series(f1_8, name='f1')
auc8[ 'F Score'] = pd.DataFrame(f1_8)



print '[+] Building AUC for model 9...'
auc9 = roc_auc_score(df9['Class'].astype(bool).values, df9['predicted'].values)
auc9 = pd.Series(auc9, name='auc')
#auc9 = pd.Series('H2O Ensemble', name='Model')
auc9 = pd.DataFrame(auc9)
auc9['model'] = 'H2O Ensemble'



print confusion_matrix(df9['Class'].astype(bool).values, df9['predicted'].values)
rec9 = recall_score(df9['Class'].astype(bool).values, df9['predicted'].values)
rec9 = pd.Series(rec9, name='rec')
auc9[ 'Recal'] = pd.DataFrame(rec9)


mcc9 = matthews_corrcoef(df9['Class'].astype(bool).values, df9['predicted'].values)
mcc9 = pd.Series(mcc9, name='mcc')
auc9[ 'MCC'] = pd.DataFrame(mcc9)

pc9 = precision_score(df9['Class'].astype(bool).values, df9['predicted'].values)
pc9 = pd.Series(pc9, name='pc')
auc9[ 'Precision'] = pd.DataFrame(pc9)

f1_9 = f1_score(df9['Class'].astype(bool).values, df9['predicted'].values)
f1_9 = pd.Series(f1_9, name='f1')
auc9[ 'F Score'] = pd.DataFrame(f1_9)

print '[+] Building AUC for model 10...'
auc10 = roc_auc_score(df10['Class'].astype(bool).values, df10['predict'].values)
auc10 = pd.Series(auc10, name='auc')
#auc10 = pd.Series('H2O Ensemble', name='Model')
auc10 = pd.DataFrame(auc10)
auc10['model'] = 'H2O RF'

print confusion_matrix(df10['Class'].astype(bool).values, df10['predict'].values)
rec10 = recall_score(df10['Class'].astype(bool).values, df10['predict'].values)
rec10 = pd.Series(rec10, name='rec')
auc10[ 'Recal'] = pd.DataFrame(rec10)

mcc10 = matthews_corrcoef(df10['Class'].astype(bool).values, df10['predict'].values)
mcc10 = pd.Series(mcc10, name='mcc')
auc10[ 'MCC'] = pd.DataFrame(mcc10)

pc10 = precision_score(df10['Class'].astype(bool).values, df10['predict'].values)
```

```
pc10 = pd.Series(pc10, name='pc')
auc10[ 'Precision'] = pd.DataFrame(pc10)

f1_10 = f1_score(df10['Class'].astype(bool).values, df10['predict'].values)
f1_10 = pd.Series(f1_10, name='f1')
auc10[ 'F Score'] = pd.DataFrame(f1_10)

print '[+] Building AUC for model 11...'
auc11 = roc_auc_score(df11['Class'].astype(bool).values, df11['predict'].values)
auc11 = pd.Series(auc11, name='auc')
#auc11 = pd.Series('H2O Ensemble', name='Model')
auc11 = pd.DataFrame(auc11)
auc11['model'] = 'H2O GLM'

print confusion_matrix(df11['Class'].astype(bool).values, df11['predict'].values)
rec11 = recall_score(df11['Class'].astype(bool).values, df11['predict'].values)
rec11 = pd.Series(rec11, name='rec')
auc11[ 'Recal'] = pd.DataFrame(rec11)

mcc11 = matthews_corrcoef(df11['Class'].astype(bool).values, df11['predict'].values)
mcc11 = pd.Series(mcc11, name='mcc')
auc11[ 'MCC'] = pd.DataFrame(mcc11)

pc11 = precision_score(df11['Class'].astype(bool).values, df11['predict'].values)
pc11 = pd.Series(pc11, name='pc')
auc11[ 'Precision'] = pd.DataFrame(pc11)

f1_11 = f1_score(df11['Class'].astype(bool).values, df11['predict'].values)
f1_11 = pd.Series(f1_11, name='f1')
auc11[ 'F Score'] = pd.DataFrame(f1_11)

print '[+] Building AUC for model 12...'
auc12 = roc_auc_score(df12['Class'].astype(bool).values, df12['predict'].values)
auc12 = pd.Series(auc12, name='auc')
#auc12 = pd.Series('H2O Ensemble', name='Model')
auc12 = pd.DataFrame(auc12)
auc12['model'] = 'Deep Learning CV'

print confusion_matrix(df12['Class'].astype(bool).values, df12['predict'].values)
rec12 = recall_score(df12['Class'].astype(bool).values, df12['predict'].values)
rec12 = pd.Series(rec12, name='rec')
auc12[ 'Recal'] = pd.DataFrame(rec12)

mcc12 = matthews_corrcoef(df12['Class'].astype(bool).values, df12['predict'].values)
mcc12 = pd.Series(mcc12, name='mcc')
auc12[ 'MCC'] = pd.DataFrame(mcc12)

pc12 = precision_score(df12['Class'].astype(bool).values, df12['predict'].values)
pc12 = pd.Series(pc12, name='pc')
auc12[ 'Precision'] = pd.DataFrame(pc12)

f1_12 = f1_score(df12['Class'].astype(bool).values, df12['predict'].values)
f1_12 = pd.Series(f1_12, name='f1')
auc12[ 'F Score'] = pd.DataFrame(f1_12)

print '[+] Building AUC for model 13...'
auc13 = roc_auc_score(df13['Class'].astype(bool).values, df13['predict'].values)
```

```
auc13 = pd.Series(auc13, name='auc')
#auc13 = pd.Series('H2O Ensemble', name='Model')
auc13 = pd.DataFrame(auc13)
auc13['model'] = 'Deep Learning Grid'

print confusion_matrix(df13['Class'].astype(bool).values, df13['predict'].values)
rec13 = recall_score(df13['Class'].astype(bool).values, df13['predict'].values)
rec13 = pd.Series(rec13, name='rec')
auc13[ 'Recal'] = pd.DataFrame(rec13)

mcc13 = matthews_corrcoef(df13['Class'].astype(bool).values, df13['predict'].values)
mcc13 = pd.Series(mcc13, name='mcc')
auc13[ 'MCC'] = pd.DataFrame(mcc13)

pc13 = precision_score(df13['Class'].astype(bool).values, df13['predict'].values)
pc13 = pd.Series(pc13, name='pc')
auc13[ 'Precision'] = pd.DataFrame(pc13)

f1_13 = f1_score(df13['Class'].astype(bool).values, df13['predict'].values)
f1_13 = pd.Series(f1_13, name='f1')
auc13[ 'F Score'] = pd.DataFrame(f1_13)

pandas_dataframe = pd.concat((auc1, auc2, auc3, auc4, auc5, auc6, auc11,auc8 , auc12, auc13, auc7, auc10 , auc9), axis=0)


# Recipe outputs
model_Scoring = dataiku.Dataset("Model_Scoring")
model_Scoring.write_with_schema(pandas_dataframe)
```