Use H2O and data.table to build models on large data sets in R

ANALYTICS VIDHYA CONTENT TEAM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/AUTHOR/AVCONTENTTEAM/), MAY 12, 2016 LO...

Introduction

Last week, I wrote an introductory article (https://www.analyticsvidhya.com/blog/2016/05/data-table-data-frame-work-large-data-sets/) on the package data.table. It was intended to provide you a head start and become familiar with its unique and short syntax. The next obvious step is to focus on modeling, which we will do in this post today.

With data.table, you no longer need to worry about your m). Atleast, I used to think of myself as a crippled R user when faced wit ould like to thank Matt Dowle (https://www.linkedin.com/in/mattdowle) again for this accomplishment.

Last week, I received an email saying:

> *Okay, I get it. data.table empowers us to do data exploration & manipulation. But, what about model building ? I work with 8GB RAM. Algorithms like random forest (ntrees = 1000) takes forever to run on my data set with 800,000 rows.*

I'm sure there are many R users who are trapped in a similar situation. To overcome this painstaking hurdle, I decided to write this post which demonstrates using the two most powerful packages i.e. H2O and data.table.

For practical understanding, I've taken the data set from a practice problem (http://datahack.analyticsvidhya.com/contest/black-friday) and tried to improve the score using 4 different machine learning algorithms (with H2O) & feature engineering (with data.table). So, get ready for a journey from rank 154th to 25th on the leaderboard.

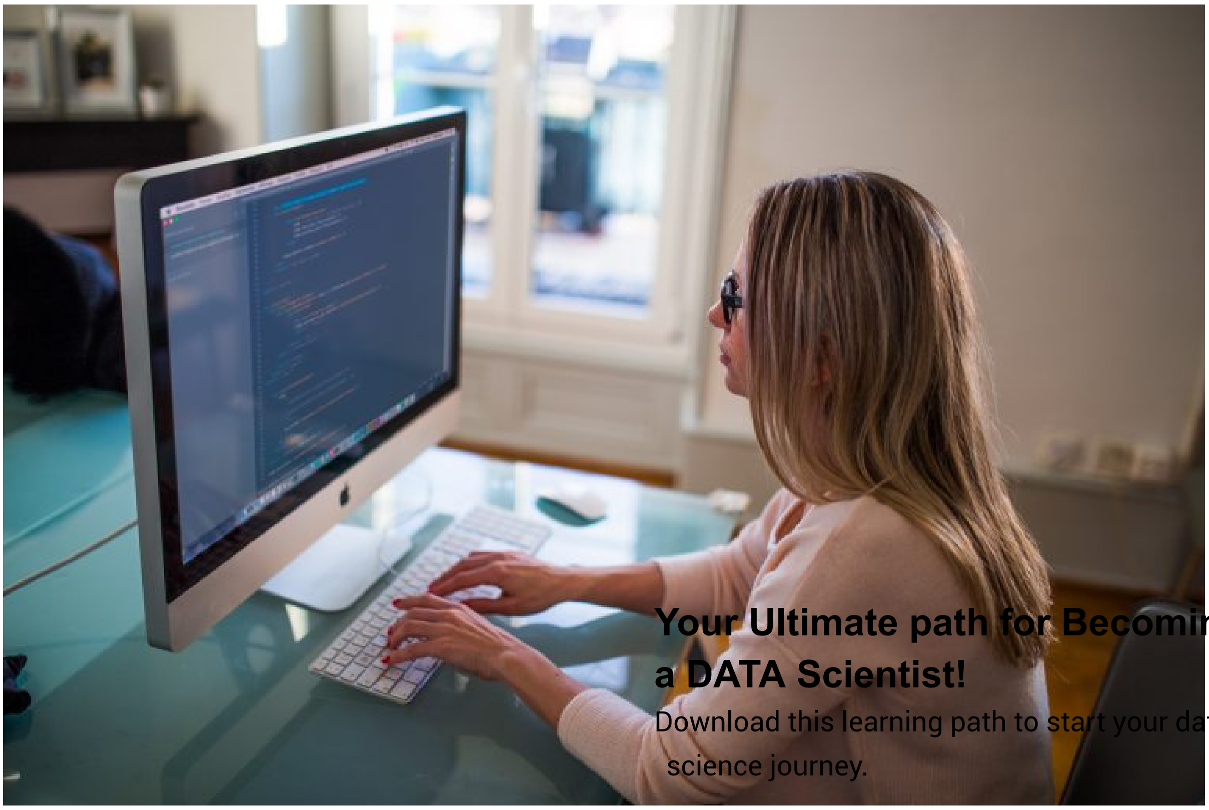


Table of Contents

1. Getting Started
2. Data Exploration using data.table and ggplot
3. Data Manipulation using data.table
4. Model Building (using H2O)
   - Regression
   - Random Forest
   - GBM
   - Deep Learning

Note: Consider this article as a starters guide for model building using data.table and H2O. I haven't explained these algorithms in details. Rather, the focus is kept on implementing these algorithms using H2O. Dont worry, links to resources are provided.

What is H2O ?

H2O (http://www.h2o.ai/) is an open source machine learning platform where companies can build models on large data sets (no sampling needed) and achieve accurate predictions. It is incredibly fast, scalable and easy to implement at any level.

In simple words, they provide a GUI driven platform to companies for doing faster data computations. Currently, their platform supports advanced & basic level algorithms such as deep learning, boosting, bagging, naive bayes, principal component analysis, time series, k-means, generalized linear models.

In addition, H2O has released APIs for R, Python, Spark, Hadoop users so that people like us can use it to build models at individual level. Needless to say, it's free to use and instigates faster computation.

What makes it faster ?

H2O has a clean and clear feature of directly connecting the tool (R or Python) with your machine's CPU. This way we get to channelize more memory, processing power to the tool for making faster computations. This will allow computations to take place at 100% CPU capacity (shown below). It can also be connected with clusters at cloud platforms for doing computations.

Along with, it uses in-memory compression to handle large data sets even with a small cluster. It also include provisions to implement parallel distributed networl

Tip: In order to channelize all your CPU's processing power for model computation, avoid using any application or software which consumes too much memory. Specially, avoid opening too many tabs on google chrome or any other web browser.

Solving a Problem

Let's get down to use these package and build some nice models.

1. Getting Started

Data Set: I've taken the data set from Black Friday Practice Problem. The data set has two parts: Train and Test. Train data set contains 550068 observations. Test data set contains 233599 observations. To download the data and read the problem statement: Click Here (https://datahack.analyticsvidhya.com/contest/black-friday/). One time login will be required.

Let's get started!

Ideally, the first step in model building is *hypothesis generation*. This step is carried out after you have read the problem statement but not seen the data.

Since, this guide isn't designed to demonstrate all predictive modeling steps, I leave that upto you. Here's a good resource to freshen up your basics: Guide to Hypothesis Generation (https://www.analyticsvidhya.com/blog/2015/09/hypothesis-testing-explained/). If you do this step, may be you could end up creating a better model than mine. Do give your best shot.

Starting with loading data in R.

```
> path <- "C:/Users/manish/desktop/Data/H2O"
> setwd(path)

#install and load the package
> install.packages("data.table")
> library(data.table)

#load data using fread
> train <- fread("train.csv", stringsAsFactors = T)
> test <- fread("test.csv", stringsAsFactors = T)
```

Within seconds, fread loads the data in R. It's that fast. The parameter stringsAsFactors ensures that character vectors are converted into factors. Let's quickly c

```
#No. of rows and columns in Train
> dim(train)
[1] 550068     12
```

```
#No. of rows and columns in Test
> dim(test)
[1] 233599     11

> str(train)
Classes ‘data.table’ and 'data.frame': 550068 obs. of 12 variables:
$ User_ID : int 1000001 1000001 1000001 1000001 1000002 1000003 1000004 1000004 1000004...
$ Product_ID : Factor w/ 3631 levels "P00000142","P00000242",..: 673 2377 853 829 2735 2632
$ Gender : Factor w/ 2 levels "F","M": 1 1 1 1 2 2 2 2 2 2 ...
$ Age : Factor w/ 7 levels "0-17","18-25",..: 1 1 1 1 7 3 5 5 5 3 ...
$ Occupation : int 10 10 10 10 16 15 7 7 7 20 ...
$ City_Category : Factor w/ 3 levels "A","B","C": 1 1 1 1 3 1 2 2 2 1 ...
$ Stay_In_Current_City_Years: Factor w/ 5 levels "0","1","2","3",..: 3 3 3 3 5 4 3 3 3 2 ..
$ Marital_Status : int 0 0 0 0 0 0 1 1 1 1 ...
$ Product_Category_1 : int 3 1 12 12 8 1 1 1 1 8 ...
$ Product_Category_2 : int NA 6 NA 14 NA 2 8 15 16 NA ...
$ Product_Category_3 : int NA 14 NA NA NA NA 17 NA NA NA ...
$ Purchase : int 8370 15200 1422 1057 7969 15227 19215 15854 15686 7871 ...
- attr(*, ".internal.selfref")=<externalptr>
```

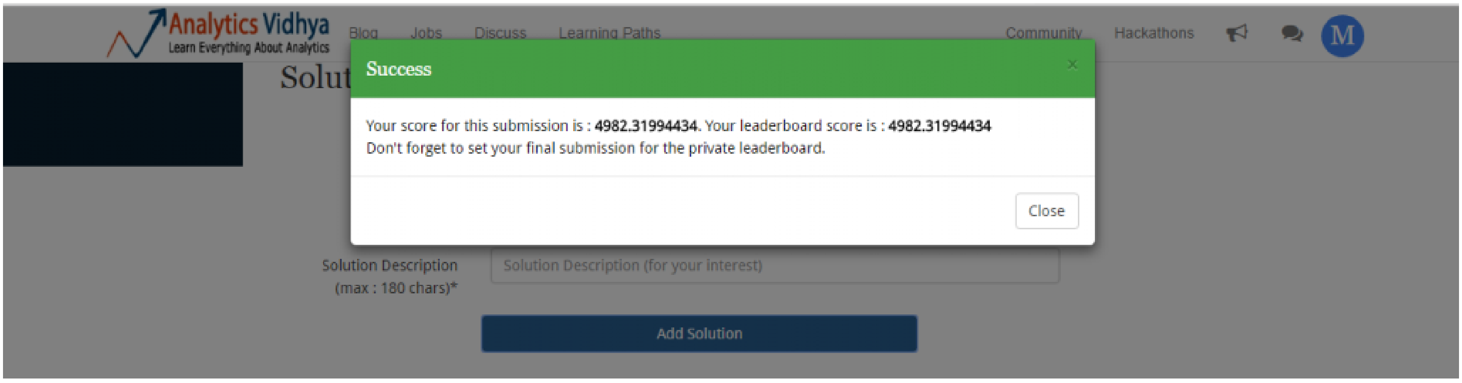
What do we see ? I see 12 variables, 2 of which seems to have so many NAs. If you have read the problem description and data information, we see *Purchase* is the dependent variable, rest 11 are independent variables.

Looking at the nature of *Purchase* variable (continuous), we can infer that this is a regression problem. Even though, the competition is closed but we can still check our score and evaluate how good we could have done. Let's make our first submission.

With all the data points we've got, we can make our first set of prediction using mean. This is because, mean prediction will give us a good approximation of prediction error. Taking this as baseline prediction, our model won't do worse than this.

```
#first prediction using mean
> sub_mean <- data.frame(User_ID = test$User_ID, Product_ID = test$Product_ID, Purchase = m
> write.csv(sub_mean, file = "first_sub.csv", row.names = F)
```

It was this simple. Now, I'll upload the resultant file and check my score and rank. Don't forget to convert .csv to .zip format before you upload. You can upload and check your solution at the competition page (http://datahack.analyticsvidhya.com/contest/black-friday-data-hack)



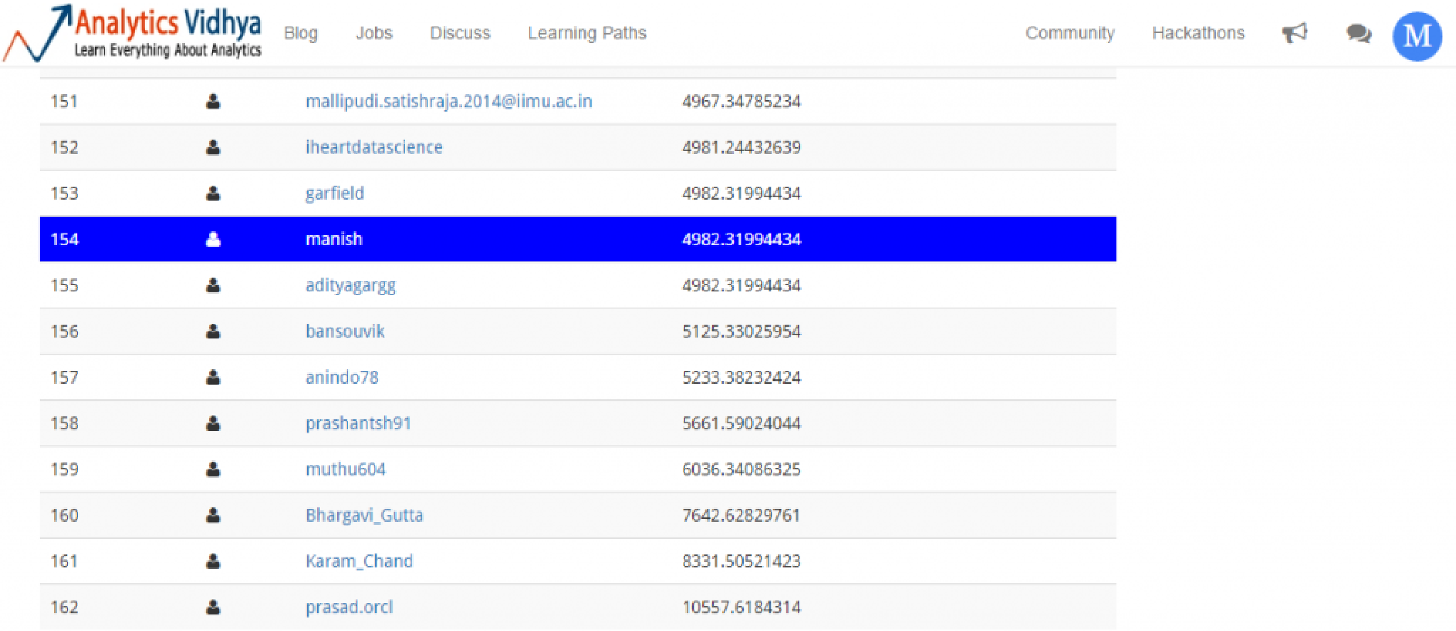
[redacted]

Success

Your score for this submission is : **4982.31994434**. Your leaderboard score is : **4982.31994434**
Don't forget to set your final submission for the private leaderboard.

Solution Description
(max : 180 chars)*

Our mean prediction gives us a mean squared error of 4982.3199. But, how good is it? Let's check my ranking on leaderboard.



151	👤	mallipudi.satishraja.2014@iimu.ac.in	4967.34785234
152	👤	iheartdatascience	4981.24432639
153	👤	garfield	4982.31994434
154	👤	manish	4982.31994434
155	👤	adityagargg	4982.31994434
156	👤	bansouvik	5125.33025954
157	👤	anindo78	5233.38232424
158	👤	prashantsh91	5661.59024044
159	👤	muthu604	6036.34086325
160	👤	Bhargavi_Gutta	7642.62829761
161	👤	Karam_Chand	8331.50521423
162	👤	prasad.orcl	10557.6184314

Thankfully, I am not last. So, mean prediction got me 154 / 162 rank. Let's improve this score and attempt to rise up the leader board.

Before starting with univariate analysis, let's quick summarize both the files (train and test) and decipher, if there exist any disparity.

```
> summary (train)

> summary (test)
```

Look carefully (check at your end) , do you see any differe one. If you carefully compare *Product_Category_1*, *Product_Category_2* & *Product_Category_3* in test and train data, there exist a disparity in *max* value. *max* value of *Product_Category_1* is 20 whereas for others is 18. These extra category levels appears to be noise. Make a note this. We'll need to remove them.

Let's combine the data set. I've used *rbindlist* function from data.table, since it's faster than *rbind*.

```
 #combine data set
 > test[,Purchase := mean(train$Purchase)]
 > c <- list(train, test)
 > combin <- rbindlist(c)
```

In the code above, we've first added the *Purchase* variable in the test set so that both data sets have equal number of columns. Now, we'll do some data exploration.

2. Data Exploration using data.table & ggplot

In this section, we'll do some univariate and bivariate analysis, and try to understand the relationship among given variables. Let's start with univariate.

```
 #analyzing gender variable
 > combin[,prop.table(table(Gender))] Gender
F          M
0.2470896 0.7529104

 #Age Variable
 > combin[,prop.table(table(Age))]
 Age
 0-17        18-25       26-35       36-45       46-50       51-55       55+
 0.02722330  0.18113944  0.39942348  0.19998801  0.08329814  0.06990724  0.03902040

 #City Category Variable
 > combin[,prop.table(table(City_Category))]
 City_Category
 A         B         C
 0.2682823 0.4207642 0.3109535

 #Stay in Current Years Variable
 > combin[,prop.table(table(Stay_In_Current_City_Years))]
 Stay_In_Current_City_Years
 0         1         2         3         4+
 0.1348991 0.3527327 0.1855724 0.1728132 0.1539825

 #unique values in ID variables
 > length(unique(combin$Product_ID))
 [1] 3677
```

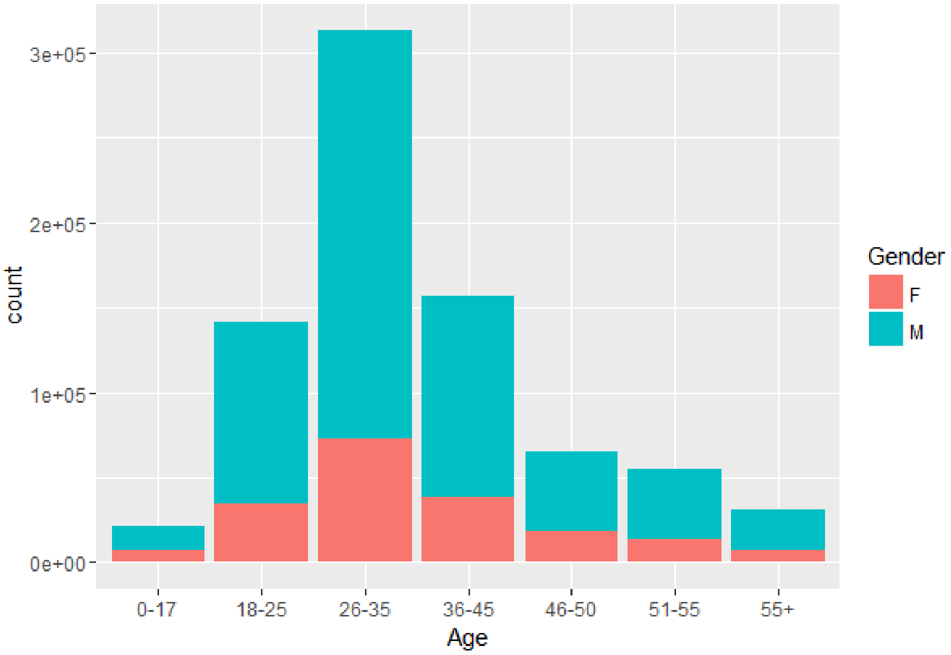
```
>length(unique(combin$User_ID))
[1] 5891

#missing values
> colSums(is.na(combin))

User_ID      Product_ID
0             0
Gender       Age
0             0
Occupation   City_Category
0             0
Stay_In_Current_City_Years   Marital_Status
0                             0
Product_Category_1           Product_Category_2
0                             245982
Product_Category_3           Purchase
545809                        0
```

Following are the inferences we can generate from univariate analysis:

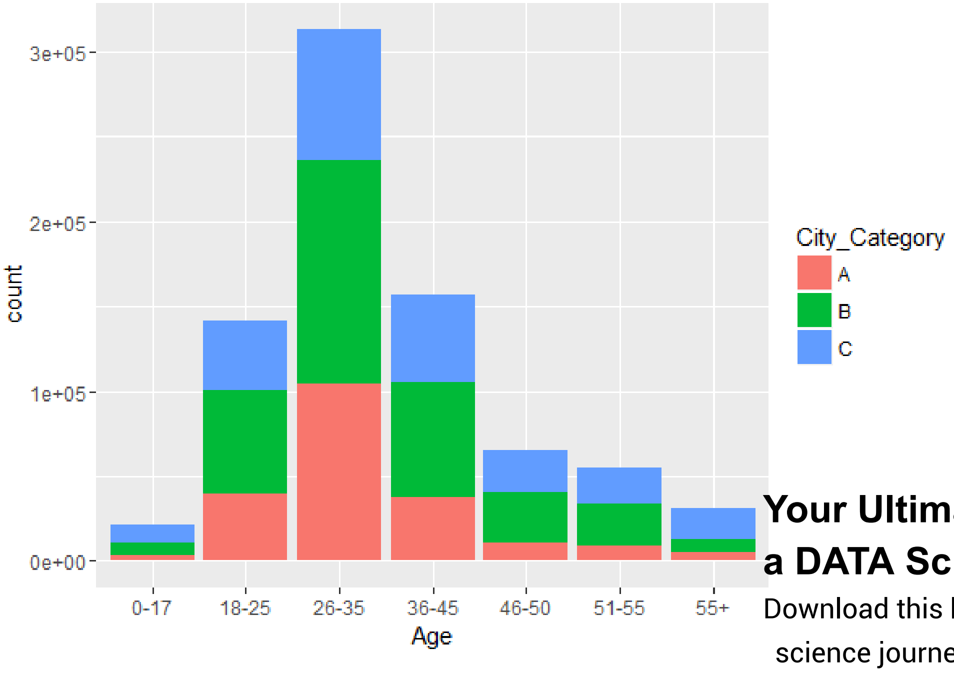
1. We need to encode *Gender* variable into 0 and 1 (good practice).
2. We'll also need to re-code the Age bins.
3. Since there are three levels in *City_Category*, we can do one-hot encoding.
4. The "4+" level of *Stay_in_Current_Years* needs to be revalued.
5. The data set does not contain all unique IDs. This gives us enough hint for feature engineering.
6. Only 2 variables have missing values. In fact, a lot of missing values, which could be capturing a hidden trend. We'll need to treat them differently.

We've got enough hints from univariate analysis. Let's tap out bivariate analysis quickly. You can always make these graphs look beautiful by adding more parameters. Here's a quick guide (https://www.analyticsvidhya.com/blog/2016/03/questions-ggplot2-package-r/) to learn making ggplots.

```
> library(ggplot2)

#Age vs Gender
> ggplot(combin, aes(Age, fill = Gender)) + geom_bar()
```



```
#Age vs City_Category
ggplot(combin, aes(Age, fill = City_Category)) + geom_bar()
```



We can also create cross tables for analyzing categorical variables. To make cross tables, we'll use the package gmodels which creates comprehensive cross tables.

```
> library(gmodels)
> CrossTable(combin$Occupation, combin$City_Category)
```

With this, you'll obtain a long comprehensive cross table of these two variables. Similarly, you can analyze other variables at your end. Our bivariate analysis haven't provided us much actionable insights. Anyways, we get to data manipulation now.

3. Data Manipulation using data.table

In this part, we'll create new variables, revalue existing variable and treat missing values. In simple words, we'll get our data ready for modeling stage.

Let's start with missing values. We saw *Product_Category_2* and *Product_Category_3* had a lot of missing values. To me, this suggests a hidden trend which can be mapped by creating a new variable. So, we'll create a new variable which will capture NAs as 1 and non-NAs as 0 in the variables *Product_Category_2* and *Product_Category_3*.

```
#create a new variable for missing values
> combin[,Product_Category_2_NA := ifelse(sapply(combin$Product_Category_2, is.na) ==    TR
> combin[,Product_Category_3_NA := ifelse(sapply(combin$Product_Category_3, is.na) ==  TRUE
```

Let's now impute the missing values with any arbitrary number. Let's take -999

```
#impute missing values
> combin[,Product_Category_2 := ifelse(is.na(Product_Category_2) == TRUE, "-999",  Product_
> combin[,Product_Category_3 := ifelse(is.na(Product_Category_3) == TRUE, "-999",  Product_
```

Before proceeding to feature engineering, lastly, we'll revalue variable levels as inferred from our univariate analysis.

```
#set column level
> levels(combin$Stay_In_Current_City_Years)[levels(combin$Stay_In_Current_City_Years) ==  "

#recoding age groups
> levels(combin$Age)[levels(combin$Age) == "0-17"] <- 0
> levels(combin$Age)[levels(combin$Age) == "18-25"] <- 1
> levels(combin$Age)[levels(combin$Age) == "26-35"] <- 2
> levels(combin$Age)[levels(combin$Age) == "36-45"] <- 3
> levels(combin$Age)[levels(combin$Age) == "46-50"] <- 4
> levels(combin$Age)[levels(combin$Age) == "51-55"] <- 5
> levels(combin$Age)[levels(combin$Age) == "55+"]

#convert age to numeric
> combin$Age <- as.numeric(combin$Age)
```

```
#convert Gender into numeric
> combin[, Gender := as.numeric(as.factor(Gender)) - 1]
```

It is advisable to convert factor variables into numeric or integer for modeling purpose.

Let's now move one step ahead, and create more new variables a.k.a feature engineering. To know more about feature engineering, you can read more (https://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/).

During univariate analysis, we discovered that ID variables have lesser unique values as compared to total observations in the data set. It means there are User_IDs or Product_IDs must have appeared repeatedly in this data set.

Let's create a new variable which captures the count of these ID variables. Higher user count suggests that a particular user has purchased products multiple times. High product count suggests that a product has been purchased many a times, which shows its popularity.

```
#User Count
> combin[, User_Count := .N, by = User_ID]
```

```
#Product Count
> combin[, Product_Count := .N, by = Product_ID]
```

Also, we can calculate the mean purchase price of a product. Because, lower the purchase price, higher will be the chances of that product being bought or vice versa. Similarly, we can create another variable which maps the average purchase price by user i.e. how much purchase (on an average) is made by a user. Let's do it.

```
#Mean Purchase of Product
> combin[, Mean_Purchase_Product := mean(Purchase), by = Product_ID]
```

```
#Mean Purchase of User
> combin[, Mean_Purchase_User := mean(Purchase), by = User_ID]
```

Now, we are only left with one hot encoding of *City_Category* variable. This can be done in one line using library *dummies*.

```
> library(dummies)
> combin <- dummy.data.frame(combin, names = c("City_Category"), sep = "_")
```

Before, proceeding to modeling stage, let's check data types of variables once, and make the required changes, if necessary.

```
#check classes of all variables
> sapply(combin, class)

#converting Product Category 2 & 3
> combin$Product_Category_2 <- as.integer(combin$Product_Category_2)
> combin$Product_Category_3 <- as.integer(combin$Product_Category_3)
```

4. Model Building using H2O

In this section, we'll explore the power of different machine learning algorithms in H2O. We'll build models with Regression, Random Forest, GBM and Deep Learning.

Make sure you don't use these algorithms like a black box. It is advisable to know how do they work. This will help you to understand the parameters used in building these models. Here are some useful resources to learn about these algorithms:

1. Regression: Starters Guide to Regression (https://www.analyticsvidhya.com/blog/2015/10/regression-python-beginners/)
2. Random Forest, GBM: Starters Guide to Tree Based Algorithms (https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/)
3. Deep Learning: Starters Guide to Deep Learning (https://www.analyticsvidhya.com/blog/2016/03/introduction-deep-learning-fundamentals-neural-networks/)

But, first things first. Let's divide the data set into test and train.

```
#Divide into train and test
> c.train <- combin[1:nrow(train),]
> c.test <- combin[-(1:nrow(train)),]
```

As discovered in beginning that the variable *Product_Category_1* in train has some noise. Let's remove it as well by selecting all rows in *Product_Category_1* upto 18, thereby dropping rows which has category level 19 & 20.

```
> c.train <- c.train[c.train$Product_Category_1 <= 18,]
```

Now, our data set is ready for modeling. Time to install H[illegible]nd package remains same. For faster computation make sure, you've closed all other applications. So, how does H2O in R work ? It's simple actually!

R uses REST API as a reference object to send functions, data to H2O. The data set is then assigned a key for future reference. H2O doesn't uses .csv data, instead it converts .csv to its own H2O instance data. You'd be surprised to know that H2O has its own functions for data manipulation too. But, data.table is no bad either.

```
> install.packages("h2o")
> library(h2o)
```

To launch the H2O cluster, write –

```
> localH2O <- h2o.init(nthreads = -1)
```

This commands tell H2O to use all the CPUs on the machine, which is recommended. For larger data sets (say > 1,000,000 rows), h2o recommends running cluster on a server with high memory for optimal performance. Once the instance starts successfully, you can also check its status using:

```
> h2o.init()

Connection successful!

R is connected to the H2O cluster:
H2O cluster uptime: 1 days 9 hours
H2O cluster version: 3.8.1.3
H2O cluster name: H2O_started_from_R_manish_vkt788
H2O cluster total nodes: 1
H2O cluster total memory: 1.50 GB
H2O cluster total cores: 4
H2O cluster allowed cores: 4
H2O cluster healthy: TRUE
H2O Connection ip: localhost
H2O Connection port: 54321
H2O Connection proxy: NA
R Version: R version 3.2.2 (2015-08-14)
```

Let's now transfer the data from R to h2o instance. It can be accomplished using as.h2o command.

```
#data to h2o cluster
> train.h2o <- as.h2o(c.train)
> test.h2o <- as.h2o(c.test)
```

Using column index, we need to identify variables to be used in modeling as follows.

```
#check column index number
> colnames(train.h2o)
[1] "User_ID"                      "Product_ID"
[3] "Gender"                       "Age"
[5] "Occupation"                   "City_Category_A"
[7] "City_Category_B"              "City_Category_C"
[9] "Stay_In_Current_City_Years"   "Marital_Status"
[11] "Product_Category_1"          "Product_Category_2"
[13] "Product_Category_3"          "Purchase"
[15] "Product_Category_2_NA"       "Product_Category_3_NA"
[17] "User_Count"                  "Product_Count"
[19] "Mean_Purchase_Product"       "Mean_Purchase_User"

#dependent variable (Purchase)
> y.dep <- 14

#independent variables (dropping ID variables)
> x.indep <- c(3:13,15:20)
```

Let's start with Multiple Regression model.

Multiple Regression in H2O

```
> regression.model <- h2o.glm( y = y.dep, x = x.indep, training_frame = train.h2o, family =

> h2o.performance(regression.model)

H2ORegressionMetrics: glm
** Reported on training data. **

MSE: 16710563
R2 : 0.3261543
Mean Residual Deviance : 16710563
Null Deviance :1.353804e+13
Null D.o.F. :545914
Residual Deviance :9.122547e+12
Residual D.o.F. :545898
AIC :10628689
```

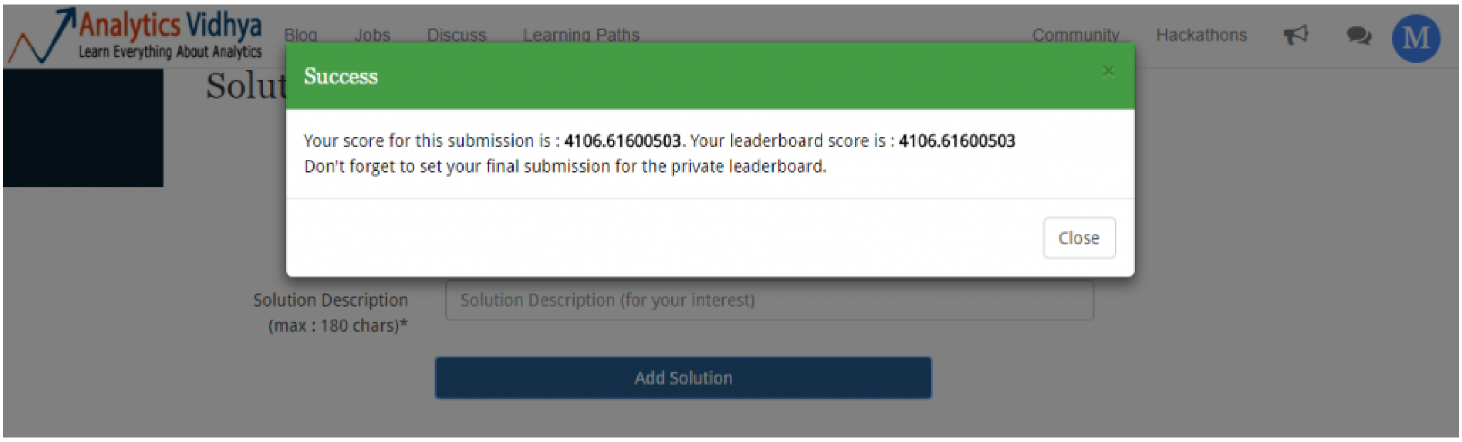
GLM algorithm in H2O can be used for all types of regression such as lasso, ridge, logistic, linear etc. A user only needs to modify the *family* parameter accordingly. For example. To do logistic regression, you can write *family = "binomial"*.

So, after we print the model results, we see that regression gives a poor R² value i.e. 0.326. It means that only 32.6% of the variance in the dependent variable is explained by independent variable and rest is unexplained. This shows that regression model is unable to capture non linear relationships.

Out of curiosity, let's check the predictions of this model. Will it be worse than mean predictions ? Let' see.

```
#make predictions
> predict.reg <- as.data.frame(h2o.predict(regression.model, test.h2o))
> sub_reg <- data.frame(User_ID = test$User_ID, Product_ID = test$Product_ID, Purchase =  p

> write.csv(sub_reg, file = "sub_reg.csv", row.names = F)
```

Let's upload the solution file (in .zip format) and check if we have got some improvement.



Success

Your score for this submission is : **4106.61600503**. Your leaderboard score is : **4106.61600503**
Don't forget to set your final submission for the private leaderboard.

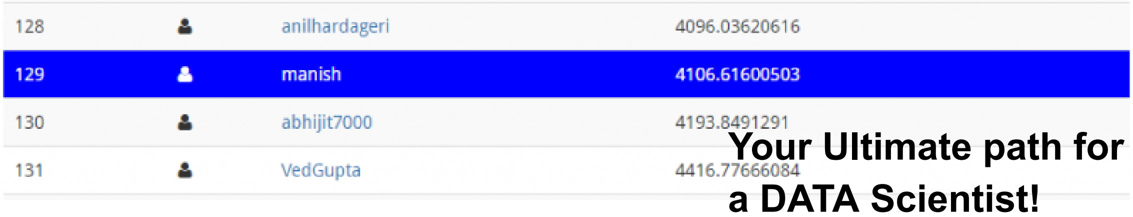
Close

Solution Description (max : 180 chars)*

Solution Description (for your interest)

Add Solution

Wow! Our prediction score has improved. We started from 4982.31 and with regression we've got an improvement over previous score. On leaderboard, this submission takes me to 129th position.



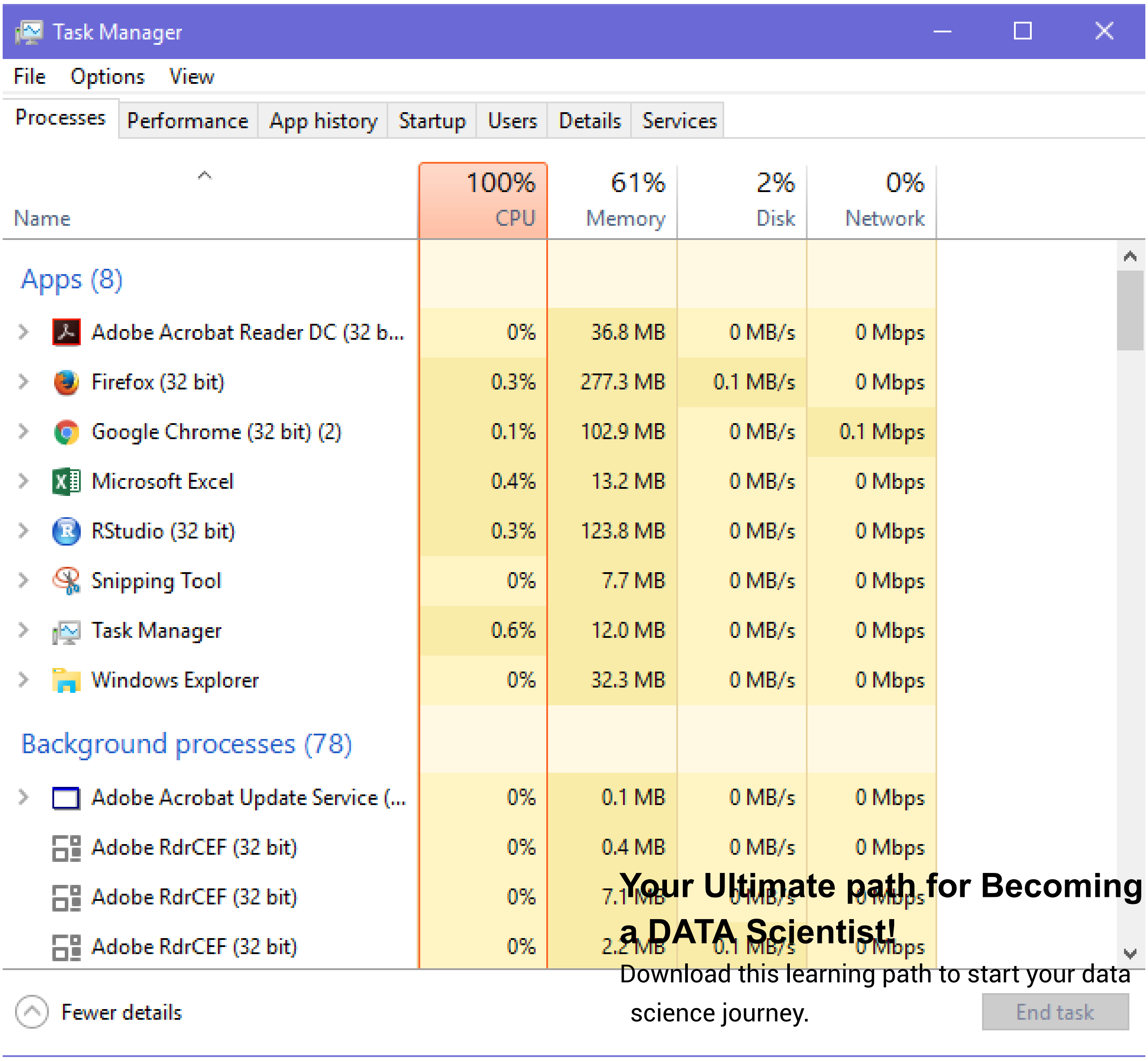
128	👤	anilhardageri	4096.03620616
129	👤	manish	4106.61600503
130	👤	abhijit7000	4193.8491291
131	👤	VedGupta	4416.77666084

It seems, we can do well if we choose an algorithm which maps non-linear relationships well. **Random Forest** is our next bet. Let's do it.

Random Forest in H2O

```
#Random Forest
> system.time(
rforest.model <- h2o.randomForest(y=y.dep, x=x.indep, training_frame = train.h2o, ntrees =
)

# |====================================================================| 100%
# user system elapsed
# 21.85 1.61 2260.33
```

With 1000 trees, random forest model took approx ~38 minutes to run. It operated at 100% CPU capacity which can be seen in Task Manager (shown below).



Name	100% CPU	61% Memory	2% Disk	0% Network
Apps (8)				
Adobe Acrobat Reader DC (32 b...	0%	36.8 MB	0 MB/s	0 Mbps
Firefox (32 bit)	0.3%	277.3 MB	0.1 MB/s	0 Mbps
Google Chrome (32 bit) (2)	0.1%	102.9 MB	0 MB/s	0.1 Mbps
Microsoft Excel	0.4%	13.2 MB	0 MB/s	0 Mbps
RStudio (32 bit)	0.3%	123.8 MB	0 MB/s	0 Mbps
Snipping Tool	0%	7.7 MB	0 MB/s	0 Mbps
Task Manager	0.6%	12.0 MB	0 MB/s	0 Mbps
Windows Explorer	0%	32.3 MB	0 MB/s	0 Mbps
Background processes (78)				
Adobe Acrobat Update Service (...	0%	0.1 MB	0 MB/s	0 Mbps
Adobe RdrCEF (32 bit)	0%	0.4 MB	0 MB/s	0 Mbps
Adobe RdrCEF (32 bit)	0%	7.1 MB	0 MB/s	0 Mbps
Adobe RdrCEF (32 bit)	0%	2.2 MB	0.1 MB/s	0 Mbps

Your model might not take same time because of difference in our machine specifications. Also, I had to open web browsers which consumed a lot of memory. Actually, your model might take lesser time. You can check the performance of this model using the same comm

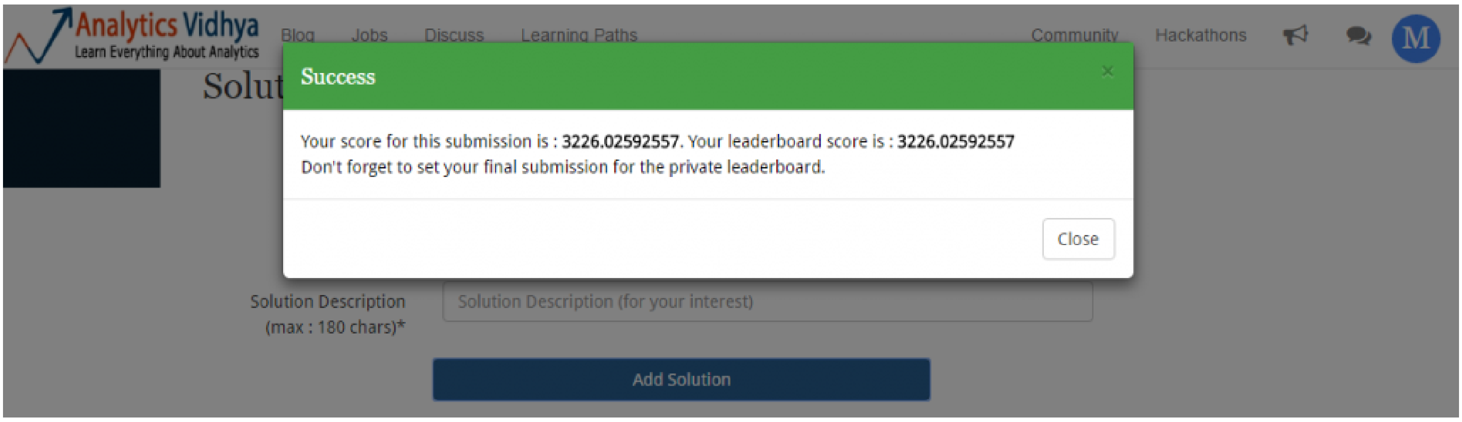
```
> h2o.performance(rforest.model)
```

```
#check variable importance
> h2o.varimp(rforest.model)
```

Let's check the leaderboard performance of this model by making predictions. Do you think our score will improve ? I'm a little hopeful, though!

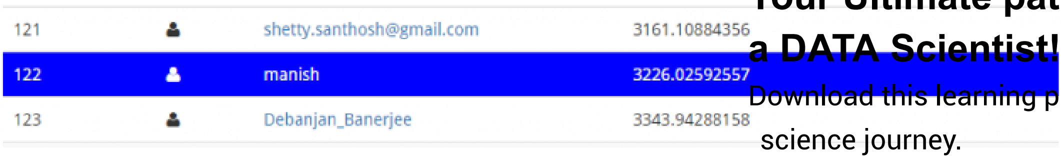
```
#making predictions on unseen data
> system.time(predict.rforest <- as.data.frame(h2o.predict(rforest.model, test.h2o)))
# |====================================================================| 100%
# user system elapsed
# 0.44 0.08 21.68

#writing submission file
> sub_rf <- data.frame(User_ID = test$User_ID, Product_ID = test$Product_ID, Purchase =  pr
> write.csv(sub_rf, file = "sub_rf.csv", row.names = F)
```

Making predictions took ~ 22 seconds. Now is the time to upload the submission file and check the results.



Your score for this submission is : 3226.02592557. Your leaderboard score is : 3226.02592557
Don't forget to set your final submission for the private leaderboard.

Random Forest was able to map non-linear relations way better than regression (as expected). With this score, my ranking on leaderboard moves to 122:



121	👤	shetty.santhosh@gmail.com	3161.10884356
122	👤	manish	3226.02592557
123	👤	Debanjan_Banerjee	3343.94288158

This gave a slight improvement on leaderboard, but not as significant as expected. May be **GBM, a boosting algorithm** can help us.

GBM in H2O

If you are new to GBM, I'd suggest you to check the resources given in the start of this section. We can implement GBM in H2O using a simple line of code:

```
#GBM
system.time(
gbm.model <- h2o.gbm(y=y.dep, x=x.indep, training_frame = train.h2o, ntrees = 1000, max_dep
)
# |====================================================================| 100%
# user system elapsed
# 7.94 0.47 739.66
```

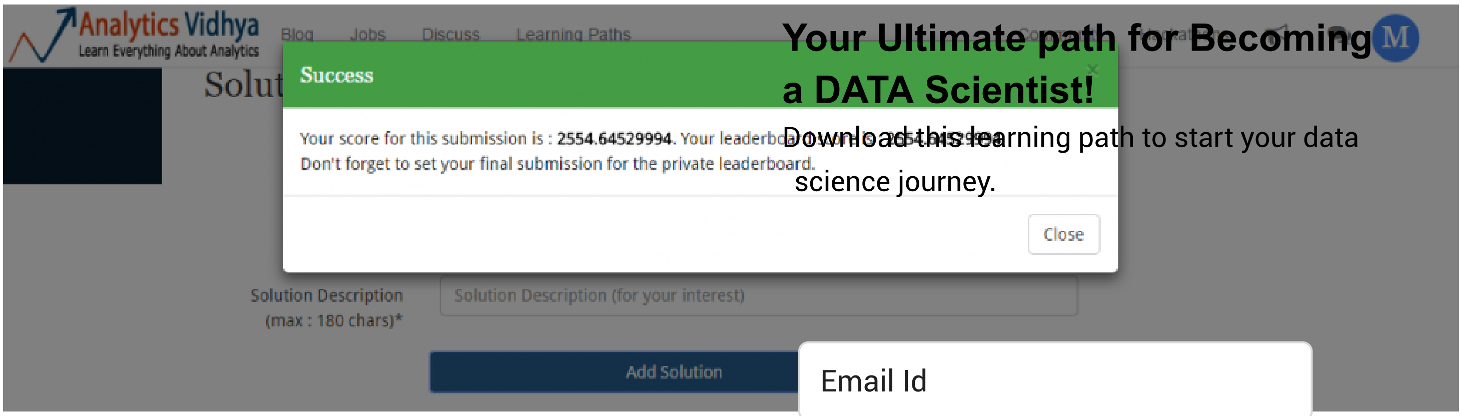
With the same number of trees, GBM took less time than random forest. It took only 12 minutes. You can check the performance of this model using:

```
> h2o.performance (gbm.model)
H2ORegressionMetrics: gbm
** Reported on training data. **
MSE: 6319672
R2 : 0.7451622
Mean Residual Deviance : 6319672
```

As you can see, our R² has drastically improved as compared to previous two models. This shows signs of a powerful model. Let's make predictions and check if this model brings us some improvement.

```
#making prediction and writing submission file
> predict.gbm <- as.data.frame(h2o.predict(gbm.model, test.h2o))
> sub_gbm <- data.frame(User_ID = test$User_ID, Product_ID = test$Product_ID, Purchase = pre
> write.csv(sub_gbm, file = "sub_gbm.csv", row.names = F)
```

We have created the submission file. Let's upload it and check if we've got any improvement.

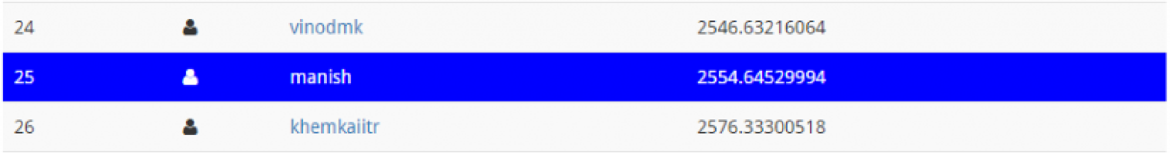


Success

Your score for this submission is : 2554.64529994. Your leaderbo
Don't forget to set your final submission for the private leaderboard.

Solution Description (max : 180 chars)* Solution Description (for your interest)

I never doubted GBM once. If done well, boosting algorithms usually pays off well. Now, will be interesting to see my leaderboard position:



24	👤	vinodmk	2546.63216064
25	👤	manish	2554.64529994
26	👤	khemkaiitr	2576.33300518

This is a massive leaderboard jump! It's like a freefall but safe landing from 122nd to 25th rank. Can we do better ? May be, we can. Let's now use **Deep Learning** algorithm in H2O and try to improve this score.

Deep Learning in H2O

Let me give you a quick overview of deep learning. In deep learning algorithm, there exist 3 layers namely input layer, hidden layer and output layer. It works as follows:

1. We feed the data to input layer.
2. It then transmits the data to hidden layer. These hidden layer comprises of neurons. These neurons uses some function and assist in mapping non linear relationship among the variables.The hidden layers are user specified.
3. Finally, these hidden layers delivers the output to output layer which then gives us the result.

Let's implement this algorithm now.

```
#deep learning models
> system.time(
             dlearning.model <- h2o.deeplearning(y = y.dep,
             x = x.indep,
             training_frame = train.h2o,
             epoch = 60,
             hidden = c(100,100),
             activation = "Rectifier",
             seed = 1122
             )
)
# |===================================| 100%
# user system elapsed
# 0.83 0.05 129.69
```

It got executed even faster than GBM model. GBM took ~730 seconds. The parameter *hidden* instructs the algorithms to create 2 hidden layers of 100 neurons each. e[illegible] passes on the train data to be carried out. *Activation* refers to the activation function to be used throughout the network.

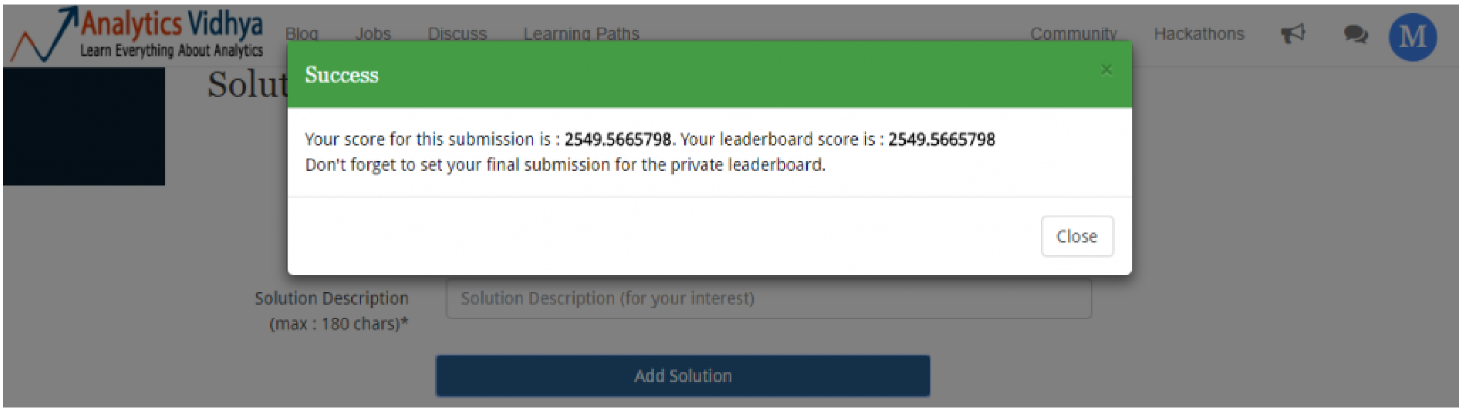
Anyways, let's check its performance.

```
> h2o.performance(dlearning.model)
H2ORegressionMetrics: deeplearning
** Reported on training data. **
MSE: 6215346
R2 : 0.7515775
Mean Residual Deviance : 6215346
```

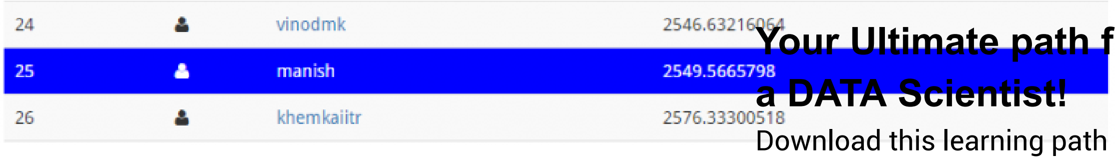
We see further improvement in the R² metric as compared to GBM model. This suggests that deep learning model has successfully captured large chunk of unexplained variances in the model. Let's make the predictions and check the final score.

```
#making predictions
> predict.dl2 <- as.data.frame(h2o.predict(dlearning.model, test.h2o))

#create a data frame and writing submission file
> sub_dlearning <- data.frame(User_ID = test$User_ID, Product_ID = test$Product_ID, Purchas
> write.csv(sub_dlearning, file = "sub_dlearning_new.csv", row.names = F)
```

Let's upload our final submission and check the score.



Your score for this submission is : **2549.5665798**. Your leaderboard score is : **2549.5665798**
Don't forget to set your final submission for the private leaderboard.



24	👤	vinodmk	2546.63216064
25	👤	manish	2549.5665798
26	👤	khemkaiitr	2576.33300518

Though, my score improved but rank didn't. So, finally we ended up at 25th rank by using little bit of feature engineering and lot of machine learning algorithms. I hope you enjoyed this journey from rank 154th to rank 25th. If you have followed me till here, I assume you'd be ready to go one step further.

What could you do to further improve this model ?

Actually, there are multiple things you can do. Here, I list them down:

1. Do parameter tuning in GBM, Deep Learning and Random Forest.
2. Use grid search for parameter tuning. H2O has a nice function h2o.grid to do this task
3. Think of creating more features which can bring new information to the model.
4. Finally, ensemble all the results to obtain a better model.

Try these steps at your end, and let me know in comments how did it turn out for you!

End Notes

I hope you enjoyed this journey with data.table and H2O. Once you become proficient at using these two packages, you'd be able to avoid a lot of obstacles which arises due to memory issues. In this article, I discussed the steps (with R codes) to implement model building using data.table and H2O. Even though, H2O itself can undertake data munging tasks, but I believe data.table is a much easy to use (syntax wise) option.

With this article, my intent was to get you started with data.table and H2O to build models. I am sure after this modeling practice you will become curious enough to take a step further and know more about these packages.

Did this article made you learn something new? Do write in the comments about your suggestions, experience or any feedback which could allow me to help you in a better way.

You can test your skills and knowledge. Check out [Live Competitions (http://datahack.analyticsvidhya.com/contest/all)](http://datahack.analyticsvidhya.com/contest/all) and compete with best Data Scientists from all over the world.

You can also read this article on Analytics Vidhya's Android APP GET IT ON Google Play

[(//play.google.com/store/apps/details?id=com.analyticsvidhya.android&utm_source=blog_article&utm_campaign=blog&pcampaignid=MKT-Other-global-all-co-prtnr-py-PartBadge-Mar2515-1)](//play.google.com/store/apps/details?id=com.analyticsvidhya.android&utm_source=blog_article&utm_campaign=blog&pcampaignid=MKT-Other-global-all-co-prtnr-py-PartBadge-Mar2515-1)

Share this:

(https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-m

(https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/?share=facebook&nb=1)

(https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/?share=twitter&nb=1)

(https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/?share=pocket&nb=1)

(https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/?share=reddit&nb=1)

Like this:

Loading...

Related Articles

(https://www.analyticsvidhya.com/blog/2016/05/data-table-data-frame-work-large-data-sets/)

data.table() vs data.frame() - Learn to work on large data sets in R (https://www.analyticsvidhya.com/blog/2016/05/data-table-data-frame-work-large-data-sets/)

May 3, 2016

In "Machine Learning"

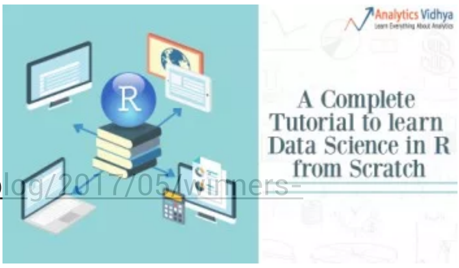
Winners solutions & approach: The QuickSolver MiniHack, DataFest 2017

(https://www.analyticsvidhya.com/blog/2017/05/winners-solutions-approach-the-quicksolver-minihack-datafest-2017/)

Winners solutions & approach: The QuickSolver MiniHack, DataFest 2017 (https://www.analyticsvidhya.com/blog/2017/05/winners-solutions-approach-the-quicksolver-minihack-datafest-2017/)

May 7, 2017

In "Machine Learning"



(https://www.analyticsvidhya.com/bl tutorial-learn-data-science-scratch/)

A Complete Tutorial to learn Data Science in R from Scratch (https://www.analyticsvidhya.com/blog/2016/02/complete-tutorial-learn-data-science-scratch/)

February 28, 2016

In "Business Analytics"

Your Ultimate path for Becoming a DATA Scientist!

Download this learning path to start your data science journey.

Email Id

Download Resource

TAGS : BAGGING (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/BAGGING/), BIVARIATE ANALYSIS (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/BIVARIATE-ANALYSIS/), BOOSTING (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/BOOSTING/), DATA EXPLORATION (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/DATA-EXPLORATION/), DATA TABLE (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/DATA-TABLE/), DEEP L (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/DEEP-LEARNING/), DEEP LEARNING IN R (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/DEEP-LEARNING-IN-R/), GBM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/GBM/), GBM IN R (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/GBM-IN-R/), H2O DEEP LEARNING (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/H2O-DEEP-LEARNING/), H2O GBM

(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/H2O-GBM/), H2O PACKAGE (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/H2O-PACKAGE/), H2O RANDOM FOREST (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/H2O-RANDOM-FOREST/), H2O REGRESSION (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/H2O-REGRESSION/), LARGE DATASET IN R (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/LARGE-DATASET-IN-R/), MACHINE LEARNING (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/MACHINE-LEARNING/), MACHINE LEARNING IN R (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/MACHINE-LEARNING-IN-R/), RANDOM FOREST IN R (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/RANDOM-FOREST-IN-R/), UNIVARIATE ANALYSIS (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/UNIVARIATE-ANALYSIS/)

NEXT ARTICLE

Business Consultant – Bangalore (2+ years of experience)

(https://www.analyticsvidhya.com/blog/2016/05/business-consultant-bangalore-2-years-experience/)

•••

PREVIOUS ARTICLE

Data Scientist (Machine Learning) – Gurgaon (2-4 years of experience)

(https://www.analyticsvidhya.com/blog/2016/05/data-scientist-machine-learning-gurgaon-2-4-years-experience/)

(https://www.analyticsvidhya.com/blog/author/avcontentteam/

Analytics Vidhya Content Team (Https://Www.Analyticsvidhya.Com/Blog/Author/Avcontentteam/)

Analytics Vidhya Content team

This article is quite old and you might not get a prompt response from the author. We request you to post this comment on Analytics Vidhya's **Discussion portal** (https://discuss.analyticsvidhya.com/) to get your queries resolved

45 COMMENTS

DECLANE

Reply

May 12, 2016 at 6:24 am (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-110864)

What's the possible best method to visualize a high dimensional data in radionics or genomics.. Any tutorial in R on high dimensional data including data reduction and data combination

ANALYTICS VIDHYA CONTENT TEAM

Reply

May 12, 2016 at 8:51 am (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-110873)

Hey Declane, you can use principal component analysis to work on high dimensional data. Check this out:
http://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-component-analysis-python/ (http://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-component-analysis-python/)
For visualization on high dimensional data, check this out:
http://www.analyticsvidhya.com/blog/2015/07/guide-data-visualization-r/ (http://www.analyticsvidhya.com/blog/2015/07/guide-data-visualization-r/)
It consists of all possible forms of visualization which you can implement in R.

VENUGOPAL

Reply

May 12, 2016 at 7:54 am (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-110869)

Really Good One … Reading huge data is what people will say problem with R But this package resolve the same …

ANALYTICS VIDHYA CONTENT TEAM

Reply

May 12, 2016 at 8:46 am (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-110872)

Thanks !

ANON

Reply

May 12, 2016 at 9:48 am (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-110875)

Nice timing, yet again! The FB recruiting competition started on Kaggle just yesterday, with a > 1GB training set.

ANALYTICS VIDHYA CONTENT TEAM Reply

May 12, 2016 at 3:38 pm (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-110893)

Glad to know! Wish you all the best for this competition. 🙂

KERN (HTTP://ME) Reply

May 12, 2016 at 9:57 am (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-110876)

I totally agree on the data.table package. It's a part of my workflow now. The syntax is clean, easy, intuitive once you get the hang of it.
Matt and Arun (creators of the data.table package) have thought out many aspects of the data munging package and added insights from their respective fields.

data.table is phenomenal for specific aspects of financial (rolling joins) and genomics data (fast overlaps) and fast too!

ANALYTICS VIDHYA CONTENT TEAM Reply

May 12, 2016 at 3:37 pm (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-110892)

Hey Kern,
Very well said. It's incredibly fast and easy to use once a user gets hold of its syntax. Cheers!

SWATY Reply

May 12, 2016 at 10:40 am (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-110877)

Thanks Manish for this article…..

Great start to h2o

HUNAIDKHAN PATHAN (HTTP://NONE) Reply

May 12, 2016 at 12:18 pm (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-110884)

Really helpful Manish, i also get out of memory error when i load dataset with more than 1000000 rows . This will make my life easy. Also great tips on how to effectively use Data.table.

Great work Manish

ANALYTICS VIDHYA CONTENT TEAM Reply

May 12, 2016 at 3:36 pm (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-110891)

Hey Hunaid, Good to know you found it helpful ! 🙂

AMEY (HTTP://WWW.CODEINVENTORY.COM) Reply

May 12, 2016 at 4:37 pm (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-110896)

Do you know whats internal of data.table ? How its manipulating or handling data so fast? Ia it taking chunk of data a time to process in-memory? Or indexing data?

AMEY (HTTP://WWW.CODEINVENTORY.COM) Reply

May 12, 2016 at 4:39 pm (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-110897)

Indeed its nice tutorial… do you know whats internal of data.table ? How its manipulating or handling data so fast? Ia it taking chunk of data a time to process in-memory? Or indexing data?

ANALYTICS VIDHYA CONTENT TEAM Reply

May 13, 2016 at 5:25 am (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-110925)

Hey Amey,
Among other features, data.table doesn't create deep copies of data sets which consumes large chunk of memory. Instead it creates shallow copies. Also, it avoids allocating memory to the intermediates steps such as filtering. It uses radix method (the fastest) for sorting. And, internally the coding is done in some form of C language which makes it faster. You should read:
http://www.analyticsvidhya.com/blog/2016/05/data-table-data-frame-work-large-data-sets/
(http://www.analyticsvidhya.com/blog/2016/05/data-table

THANISH Reply

May 16, 2016 at 4:53 am (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-111031)

HI Manish first of all it's a great article thanks for that, i have a question which is not exactly related to h20.You converted the gender variable to 1 and 0 and changed it to numeric,
age bins to 0-6 and converted to numeric,where as city category you left it to be
A,B,C and did one hot encoding on it. Why so only on city category ?
You could have done the one hot encoding for Gender as well right?(My guess is you have changed it to binary format 0,1 which is what one hot encoding does so you left it.correct me if i am wrong).
But why was one hot encoding was not done on age-bins
and left it to be 1-6 numerics. Any specific reason ?

AMINE TEFFAL Reply

May 16, 2016 at 10:58 am (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-111046)

Hi Manish,
I tried to download data but the web page containing it is disabled. Is there another way to get data.
Thanks.

JAMES Reply

May 18, 2016 at 2:52 pm (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-111137)

I've 2 variables as an linear equation lm(y ~ x, data=actual_data) with r-squared about 86%. From here I made use of the coefficients to predict the NEXT value (single value)

I use gbm from h20 and r-square about 92%. I do not wish to use TEST data. (summary(gbm.model) does not give the info in need (except r-squared)

How can I use gbm to predict a single value -, example are there similar coefficients to linear regression model ?

predict.gbm this equation, how do I change the test.h20 as a single value ?

ANALYTICS VIDHYA CONTENT TEAM Reply

July 27, 2016 at 4:35 am (https://www.analyticsvidhya.com/ ge-data-sets/#comment-114099)

Hey James,
If I have understood correctly you can predict single value using this:
predict.gbm <- as.data.frame(h2o.predict(gbm.model, data = data.frame(x = 20)) Assuming, you model has just one independent variable x. x = 20 is an hypothetical value, you can pass any.

JAMES

Reply

July 27, 2016 at 2:36 pm (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-114134)

below will work
========================

```
mynewdata<- data.frame(x=20)

#convert to h20 frame – need to perform this step otherwise cannot work
result_h20frame <- as.h2o(mynewdata)

predict.gbm <- as.data.frame(h2o.predict(gbm.model, result_h20frame))
```

JAMES

Reply

May 18, 2016 at 2:53 pm (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-111138)

```
predict.gbm <- as.data.frame (h2o.predict(gbm.model, test.h20))
```

IVANOBOTH

Reply

May 23, 2016 at 2:51 pm (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-111343)

Manish!!,
Awesome article once again, extremely helpful! I missed the chance to download the data set but i'll keep my eyes more open next time around, thanks again for sharing your knowledge and passion- the awesomness is infectious!

ANALYTICS VIDHYA CONTENT TEAM

Reply

July 27, 2016 at 4:31 am (https://www.analyticsvidhya.com/ ge-data-sets/#comment-114098)

Hi Ivan,

You can access the data set here: http://datahack.analyticsvidhya.com/contest/black-friday (http://datahack.analyticsvidhya.com/contest/black-friday)

JAMES Reply

May 27, 2016 at 3:42 pm (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-111526)

How do I retrieve the field for R-square shown in the GBM model ?

AMINE Reply

May 31, 2016 at 11:56 pm (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-111669)

How to perform K-fold cross validation with h2o ??

ANALYTICS VIDHYA CONTENT TEAM Reply

July 27, 2016 at 4:30 am (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-114097)

Hey Amine,
Most algorithms in h2o comes with a parameter `nfolds` using which you can perform cross validation.
Later you can check the cross validation performance using
`yourmodelname$model@crossvalidationmetrics`

SRAVAN Reply

June 6, 2016 at 6:05 pm (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-111933)

This is awesome!!! I really appreciate the time and effort kept into this. Could you please provide any variable reduction techniques using h2o package if available??

ANALYTICS VIDHYA CONTENT TEAM Reply

July 21, 2016 at 5:03 am (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-113753)

H2o supports Principal Component Analysis for variable reduction. You can access the function using `h2o.prcomp` and it's quite advanced than the base prcomp function.

JAMES

Reply

July 27, 2016 at 2:37 pm (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-114135)

any example of using h20.prcomp ? Just a few liners will do …

SUNNYSAI

Reply

July 19, 2016 at 11:03 am (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-113672)

Awsme Article…

Can we convert the h2o models into PMML?

ANGELA LI

Reply

July 21, 2016 at 2:34 am (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-113750)

hello.
in your section 3, data manipulation. there's one line of code that converts age from levels to numeric:
> combin$Age <- as.numeric(combin$Age)
however, this will cause all age values to become NAs (or at least when i view the dataset again all elements in age column are shown as NAs)
is this a mistake?
Great article btw! 🙂

AISHA

Reply

July 26, 2016 at 6:08 pm (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-114069)

hello.
i am following your tutorial, which is awesome btw, but my h2o glm is running for over 5 min now. is it supposed to take that long? or is my computer just blanking out?
thanks.

ANALYTICS VIDHYA CONTENT TEAM

Reply

July 27, 2016 at 4:27 am (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-114096)

Hey Thanks!
h2o.glm taking time is weird. I need to know few things to help you out. What is your system config ? What is the data set size and dimension?
Probably, this is due to memory issues.

AISHA Reply

July 28, 2016 at 5:46 pm (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-114210)

Hi Manish. Thank you for replying!
I figured out my mistake; it was indeed memory issues. Thank you!

SANTOSH Reply

July 28, 2016 at 7:13 pm (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-114215)

Hello Manish,
i am trying to follow your tutorial (which is awesome btw!), and i am stuck on section 3.
you converted age from levels to numeric using:
> combin$Age <- as.numeric(combin$Age)
but i am getting NA's for all the age values when i do run this line of code. i checked the data, and it is fine in the previous lines of code, which converts all age bins into levels.
what is happening here? thank you!!!

ANON Reply

July 31, 2016 at 1:18 pm (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-114300)

Awesome Article !! Helped me a lot

DIVYA Reply

August 3, 2016 at 6:24 pm (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-114416)

Hi Manish,
How do you use assemble method for continuous output?
Do you have tutorials on assembling methods?
Thank you!

AKASH RAMKUMAR Reply

August 4, 2016 at 6:21 am (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-114436)

Hey Manish,

Is Ensembling done only by taking the average of all the models or using weighted average or by allocating a percentage point to each model?

VIKAS Reply

September 9, 2016 at 8:30 am (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-115842)

Hi Manish… learnt a lot about handling large data sets from the article…thanks so much.

One query, how would the model code change if this was a classification problem? will we have to add classification=TRUE in the syntax. Or is there some other code.

MAIIA BAKHOVA (HTTP://MYABAKHOVA.BLOGSPOT.COM/) Reply

September 10, 2016 at 3:35 pm (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-115904)

Just want to mention that the line

install.packages("h2o")

does not install the package. You need to go to the H2O site and follow their procedure.

NAZIR Reply

October 5, 2016 at 8:39 am (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-116813)

Hi manish,

I am getting a "ERRR on field: _response: Response cannot be constant." while running h2o. Can you help figure out the issue please?

thanks

ABHISHEK SINGH RATHORE Reply

November 11, 2016 at 2:27 am (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-118236)

Amazing article and really great way to teach each step 🙂 thank a lot 🙂

AJAS Reply

November 29, 2016 at 6:57 am (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-118877)

Thanks man, so awesome article, i was searching for such article on H2o….:)..

DEEPU Reply

December 11, 2017 at 3:13 pm (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-146999)

Excellent article. I have become a big fan of data.table & H2O packages after going through this article. Thanks again. Keep writing.

DIPANJAN CHOWDHURY Reply

April 5, 2018 at 1:21 am (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-152376)

gbm.model=h2o.gbm(y=dep,x=indep,training_frame = train.h2o,ntrees = 1000,max_depth = 50,learn_rate = 0.01,seed = 1122)
why im havibg this error

gbm.model=h2o.gbm(y=dep,x=indep,training_frame = train.h2o,ntrees = 1000,max_depth = 50,learn_rate = 0.01,seed = 1122)

Error in FUN(X[[i]], …) : Cannot select row or column 0
In addition: Warning messages:
1: In if (is.character(x)) { :
the condition has length > 1 and only the first element will be used
2: In if (is.numeric(sel)) { :
the condition has length > 1 and only the first element will be used

---

AISHWARYA SINGH Reply

April 6, 2018 at 5:47 pm (https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/#comment-152421)

Hi Dipanjan,

The article “Use H2O and data.table to build models on large data sets in R” is quiet old now and you might not get a prompt response from the author.

I would request you to post your queries on the discuss portal (https://discuss.analyticsvidhya.com/) to get them resolved.

POPULAR POSTS

6 Useful Programming Languages for Data Science You Should Learn (that are not R and Python) (https://www.analyticsvidhya.com/blog/2019/06/6-useful-programming-languages-data-science-r-python/)

24 Ultimate Data Science Projects To Boost Your Knowledge and Skills (& can be accessed freely) (https://www.analyticsvidhya.com/blog/2018/05/24-ultimate-data-science-projects-to-boost-your-knowledge-and-skills/)

Commonly used Machine Learning Algorithms (with Python and R Codes) (https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/)

A Complete Python Tutorial to Learn Data Science from Scratch (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/)

7 Regression Techniques you should know! (https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/)

6 Powerful Open Source Machine Learning GitHub Repositories for Data Scientists (https://www.analyticsvidhya.com/blog/2019/07/6-powerful-open-source-machine-learning-github-repositories-data-scientists/)

Stock Prices Prediction Using Machine Learning and Deep Learning Techniques (with Python codes) (https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/)

Understanding Support Vector Machine algorithm from examples (along with code) (https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/)

RECENT POSTS

10 Powerful Applications of Linear Algebra in Data Science (with Multiple Resources) (https://www.analyticsvidhya.com/blog/2019/07/10-applications-linear-algebra-data-science/)

JULY 23, 2019

Computer Vision Tutorial: Implementing Mask R-CNN for Image Segmentation (with Python Code) (https://www.analyticsvidhya.com/blog/2019/07/computer-vision-implementing-mask-r-cnn-image-segmentation/)

JULY 22, 2019

Introduction to PyTorch-Transformers: An Incredible Library for State-of-the-Art NLP (with Python code) (https://www.analyticsvidhya.com/blog/2019/07/pytorch-transformers-nlp-python/)

JULY 18, 2019

How to Get Started with NLP – 6 Unique Methods to Perform Tokenization (https://www.analyticsvidhya.com/blog/2019/07/how-get-started-nlp-6-unique-ways-perform-tokenization/)

JULY 18, 2019

-

Your Ultimate path for Becoming a DATA Scientist!

Download this learning path to start your data science journey.

Email Id

Download Resource

Analytics Vidhya
ONLINE COURSE
Natural Language Processing Using Python
Learn to Solve
Text Classification Problems Using NLP

(https://courses.analyticsvidhya.com/courses/natural-language-processing-nlp?utm_source=Sticky_banner1&utm_medium=display&utm_campaign=NLPcourse)

(https://courses.analyticsvidhya.com/courses/computer-vision-using-deep-learning-version2/?utm_source=Sticky_banner1&utm_medium=display&utm_campaign=CVcourse)

ANALYTICS VIDHYA

About Us (http://www.analyticsvidhya.com/about-me/)

Our Team (https://www.analyticsvidhya.com/about-me/team/)

Career (https://www.analyticsvidhya.com/career-analytics-vidhya/)

Contact Us (https://www.analyticsvidhya.com/contact/)

Write for us (https://www.analyticsvidhya.com/about-me/write/)

DATA SCIENTISTS

Blog (https://www.analyticsvidhya.com/blog/)

Hackathon (https://datahack.analyticsvidhya.com/)

Discussions (https://discuss.analyticsvidhya.com/)

Apply Jobs (https://www.analyticsvidhya.com/jobs/)

Leaderboard (https://datahack.analyticsvidhya.com/users/)

COMPANIES

Post Jobs (https://www.analyticsvidhya.com/corporate/)

(https://trainings.analyticsvidhya.com)

Hiring Hackathons (https://datahack.analyticsvidhya.com/)

Advertising (https://www.analyticsvidhya.com/contact/)

Reach Us (https://www.analyticsvidhya.com/contact/)

JOIN OUR COMMUNITY :

f (https://www.facebook.com/AnalyticsVidhya/) 46336 Followers

(https://twitter.com/analyticsvidhya) 21461 Followers

G+ (https://plus.google.com/+Analyticsvidhya) Followers

in (https://in.linkedin.com/company/analytics-vidhya) Followers

Your Ultimate path for Becoming a DATA Scientist!

Download this learning path to start your data science journey.

Email Id

>

Download Resource