INTERNET ADVERTISEMENTS

1)Challenges faced while cleansing and pre-processing the data

a) Replacing the "?" values by NA in continuous data i.e for the first three columns
b) Estimating and Replacing the outliers for continuous data by NA's
c) Replacing all the NA's by appropriate values in continuous data (we are not simply deleting the records as suggested in the pdf)
d) Dimensionality reduction since the number of features for this dataset is huge

NOTE: To gain the domain knowledge of this particular dataset we have referred this Article called Learning to remove Internet Advertisements by Nicholas Kushmerick. By studying this article we gained knowledge of how the instances of this dataset are encoded and recorded. We also came to know in what way each feature describes and Advertisement. For eg: the local feature describes whether the URL of anchor tag and URL of image tag are present in the same server. The link for this article is mentioned in the references section at the end.

The following steps are followed in overcoming the challenge (a)

- ✓ The data type of the continuous variables are given in factor format
- ✓ We converted the factor format of these continuous variables to character and later converted the character format to numeric
- ✓ By doing this the values such as "?" are automatically converted to NA's
- ✓ By the end of this step we had few NA's in the first three columns i.e. Height, Width and Aspect Ratio

The following steps are followed in overcoming the challenge (b)

- ✓ To estimate the outliers of a particular data the domain knowledge of that particular data is required
- ✓ Here we need to estimate the outliers for Height, Width and Aspect Ratio of images used in Internet Advertisements
- ✓ For this we were able to browse and get the information regarding standard sizes of images used in Internet Advertisements
- ✓ http://www.fileformat.info/tip/web/imagesize.html
- ✓ The above mentioned link serves as a reference for finding out the standard sizes of images used in Internet Ads
- ✓ From this information we were able to find out the max and min values that can exist for an image of Internet Advertisement
- ✓ We noticed that an image can have the following max and min values for Height, Width, Aspect Ratio when used as an internet advertisement

  Height-→ Max=600, Min=30
  Width--→Max=728, Min=88

  Note: Aspect Ratio = Width/Height
  Hence We can find out the Max and Min Values of Aspect Ratio by above formula

Aspect Ratio→ Max=24.26, Min=0.14

- ✓ After we were able to get the above ranges for Height, Width and Aspect Ratio then we wrote an algorithm that could detect data which fall outside these ranges as outliers
- ✓ Finally these outliers that fall outside the ranges are replaced by NA's

<u>The following steps are followed in overcoming the challenge (c)</u>

- ✓ Now we have an accumulated set of NA values from step (a) and step (b)
- ✓ We assumed that replacing these values of NA by Mode of the observations would be more appropriate and meaningful than replacing them with Mean of the observations.
- ✓ We made the above assumption because each record represents different features of an advertisement and each record is independent of the other.
- ✓ Since this is not a time series data , we thought making a mean or a median wouldn't make much of a sense
- ✓ As this data represents the observations of a set of AD's and Non AD's we thought going for a Mode would be better option.
- ✓ By choosing Mode the NA's in Height would be replaced by frequently occurring heights of images and similarly for NA's in width would be replaced by frequently occurring widths of images.
- ✓ There was no inbuilt function like mean or median for calculating the mode. Hence we created a function on our own which could implement the functionality of Mode.
- ✓ This function is built in such a way that it could handle data of numeric, character and factor format.
- ✓ It also ignores NA's present while calculating the Mode

<u>The following steps are followed in overcoming the challenge (d)</u>

- ✓ This was probably the most biggest challenge we have faced in this problem
- ✓ The number if attributes or dimensions or features or variables for this dataset is huge. To be precise there are 1559 attributes.
- ✓ Reducing the size of attributes is important as it would avoid overfitting and also helps in building more accurate classification models which could help in capturing meaning information
- ✓ As these variables are highly correlated we used PCA to transform these large set of variables to a smaller set of variables
- ✓ We implement PCA by calling a function called preProcess and applying a transformation called Box-Cox in that function
- ✓ For understanding and implementing PCA we made use of the following link or reference
- ✓ **http://www.r-bloggers.com/computing-and-visualizing-pca-in-r/**

- We have implemented PCA for clusters of features such as URL, origurl, capture, ALT and Anchor
- After implementing the PCA in clusters of these , the total number of variables came down to **498** from **1559**. This is a significant decrease in attribute size.
- **NOTE:** The PCA can be implemented to the entire set of variables insead of applying it to clusters of variables but that wouldn't be an appropriate approach. As the variables are more correlated in clusters it is better to implement PCA in clusters of variables.
- **NOTE:** We tried implementing the PCA for the entire set of variables then the number of variables reduced to **300 from 1559.**
- But if we follow the above approach then in the later stages the performance metrics of classification models are not that good compared to the approach where we have done PCA in clusters.
- Hence we decided to choose the first approach where we implement PCA in clusters of variables such as URL Terms, origurl terms, caption etc.
- By doing this our final number of attributes in cleansed and pre-processed data came down to **498**-excluding the Ad class variable.

2) CLASSIFICATION MODEL-CLASSIFICATION TREES

a) Tree built without pruning

When the model is built without pruning the tree then the overall error rate is more than the overall error rate of the model which is pruned.

```
ConfusionMatrix_Tree_NotPruined
          Ad_Class.test
tree.pred  ad. nonad.
   ad.     189     37
   nonad.   35   1379
```

As you can see above in the confusion matrix, the model which is not pruined is predicting 189 AD cases correctly as "ad" and it is predicting 37 AD cases wrongly as "nonad"

Similarly it is predicting 1379 NONAD cases correctly as "nonad" and it is predicting 35 NONAD cases wrongly as "ad"

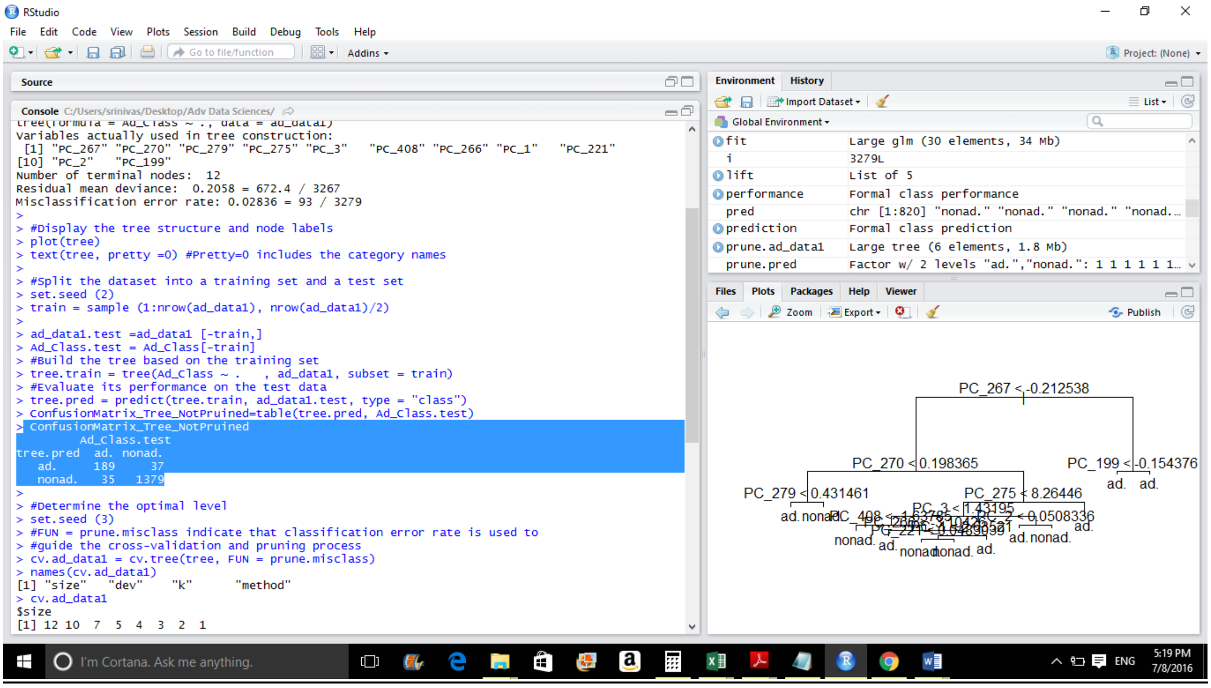
The overall error rate for the tree model which is not pruined= ((35+37)/1640)*100

(a) Overall error rate_TreeNotPruined=4.3%

Error rate and accuracy in prediciting each individual class

- Error rate in Predicting Ad class= 37/226= 16.3%
- Accuracy in Predicting Ad class= 189/226= 83.6%
- Error rate in Predicitng NonAD class= 35/1414= 2.4%
- Accuracy in Predicitng NonAD class= 1379/1414= 97.5%

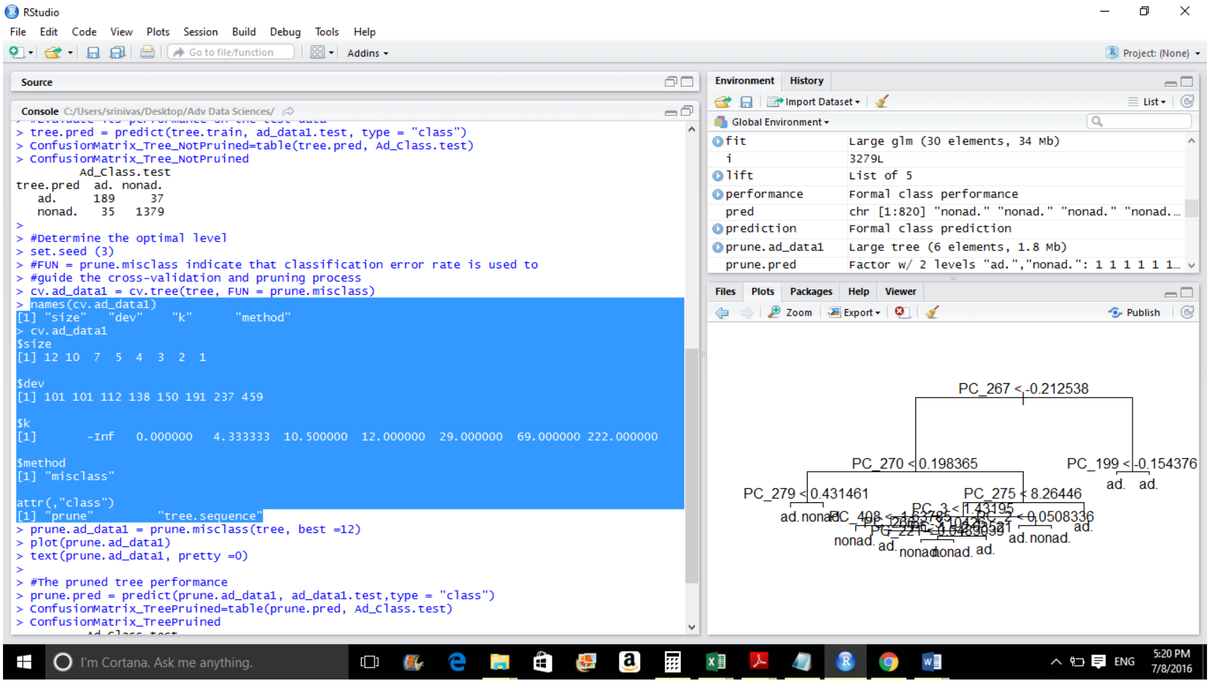
This tree model which is not pruned is more accurate in predicting NonAD classes.



```
tree(formula = Ad_Class ~ ., data = ad_data1)
Variables actually used in tree construction:
 [1] "PC_267" "PC_270" "PC_279" "PC_275" "PC_3"   "PC_408" "PC_266" "PC_1"   "PC_221"
[10] "PC_2"   "PC_199"
Number of terminal nodes:  12
Residual mean deviance:  0.2058 = 672.4 / 3267
Misclassification error rate: 0.02836 = 93 / 3279
>
> #Display the tree structure and node labels
> plot(tree)
> text(tree, pretty =0) #Pretty=0 includes the category names
>
> #Split the dataset into a training set and a test set
> set.seed (2)
> train = sample (1:nrow(ad_data1), nrow(ad_data1)/2)
>
> ad_data1.test =ad_data1 [-train,]
> Ad_Class.test = Ad_Class[-train]
> #Build the tree based on the training set
> tree.train = tree(Ad_Class ~ .    , ad_data1, subset = train)
> #Evaluate its performance on the test data
> tree.pred = predict(tree.train, ad_data1.test, type = "class")
> ConfusionMatrix_Tree_NotPruined=table(tree.pred, Ad_Class.test)
> ConfusionMatrix_Tree_NotPruined
         Ad_Class.test
tree.pred  ad. nonad.
   ad.     189     37
   nonad.   35   1379
>
> #Determine the optimal level
> set.seed (3)
> #FUN = prune.misclass indicate that classification error rate is used to
> #guide the cross-validation and pruning process
> cv.ad_data1 = cv.tree(tree, FUN = prune.misclass)
> names(cv.ad_data1)
[1] "size"   "dev"    "k"      "method"
> cv.ad_data1
$size
[1] 12 10  7  5  4  3  2  1
```

```
fit              Large glm (30 elements, 34 Mb)
  i              3279L
lift             List of 5
performance      Formal class performance
  pred           chr [1:820] "nonad." "nonad." "nonad." "nonad....
prediction       Formal class prediction
prune.ad_data1   Large tree (6 elements, 1.8 Mb)
  prune.pred     Factor w/ 2 levels "ad.","nonad.": 1 1 1 1 1 1...
```

In the above picture the highlighted blue part shows the confusion matrix of the tree model which is **not pruned**.



```
> #Evaluate its performance on the test data
> tree.pred = predict(tree.train, ad_data1.test, type = "class")
> ConfusionMatrix_Tree_NotPruined=table(tree.pred, Ad_Class.test)
> ConfusionMatrix_Tree_NotPruined
         Ad_Class.test
tree.pred  ad. nonad.
   ad.     189     37
   nonad.   35   1379
>
> #Determine the optimal level
> set.seed (3)
> #FUN = prune.misclass indicate that classification error rate is used to
> #guide the cross-validation and pruning process
> cv.ad_data1 = cv.tree(tree, FUN = prune.misclass)
> names(cv.ad_data1)
[1] "size"   "dev"    "k"      "method"
> cv.ad_data1
$size
[1] 12 10  7  5  4  3  2  1

$dev
[1] 101 101 112 138 150 191 237 459

$k
[1]       -Inf   0.000000   4.333333  10.500000  12.000000  29.000000  69.000000 222.000000

$method
[1] "misclass"

attr(,"class")
[1] "prune"          "tree.sequence"
> prune.ad_data1 = prune.misclass(tree, best =12)
> plot(prune.ad_data1)
> text(prune.ad_data1, pretty =0)
>
> #The pruned tree performance
> prune.pred = predict(prune.ad_data1, ad_data1.test,type = "class")
> ConfusionMatrix_TreePruined=table(prune.pred, Ad_Class.test)
> ConfusionMatrix_TreePruined
```

```
fit              Large glm (30 elements, 34 Mb)
  i              3279L
lift             List of 5
performance      Formal class performance
  pred           chr [1:820] "nonad." "nonad." "nonad." "nonad....
prediction       Formal class prediction
prune.ad_data1   Large tree (6 elements, 1.8 Mb)
  prune.pred     Factor w/ 2 levels "ad.","nonad.": 1 1 1 1 1 1...
```

The above picture which is highlighted (Cross-Validation) gives the size: number of terminal nodes of each tree considered, k: the value of cost complexity parameter

b) Tree built after Pruning

Note : the number of terminal nodes were 12. Using Cross validation function checked the number of terminal nodes, the value of cost complexity (k)

When the model is built after pruning the tree then the overall error rate has decreased .

```
 ConfusionMatrix_TreePruined
          Ad_Class.test
prune.pred  ad. nonad.
    ad.     192     15
    nonad.   32   1401
```

As you can see above after the tree is pruned the number of nonad cases predicted correctly as nonad cases has increased(1379 to 1401) compared to the previous tree model without pruning.

In the previous tree model without pruning it predicted only 1379 nonad cases correctly as nonad

But whereas in the current model after the tree is pruned 1401 nonad cases are correctly predicted as nonad

Similarly the number of ad cases predicted correctly as ad cases also increased from 189 to 192

Hence this model which is pruned increases the accuracy of prediction of individual classes i.e Ad and NonAd

The overall error rate for the tree model which is pruned= ((32+15)/1640)*100

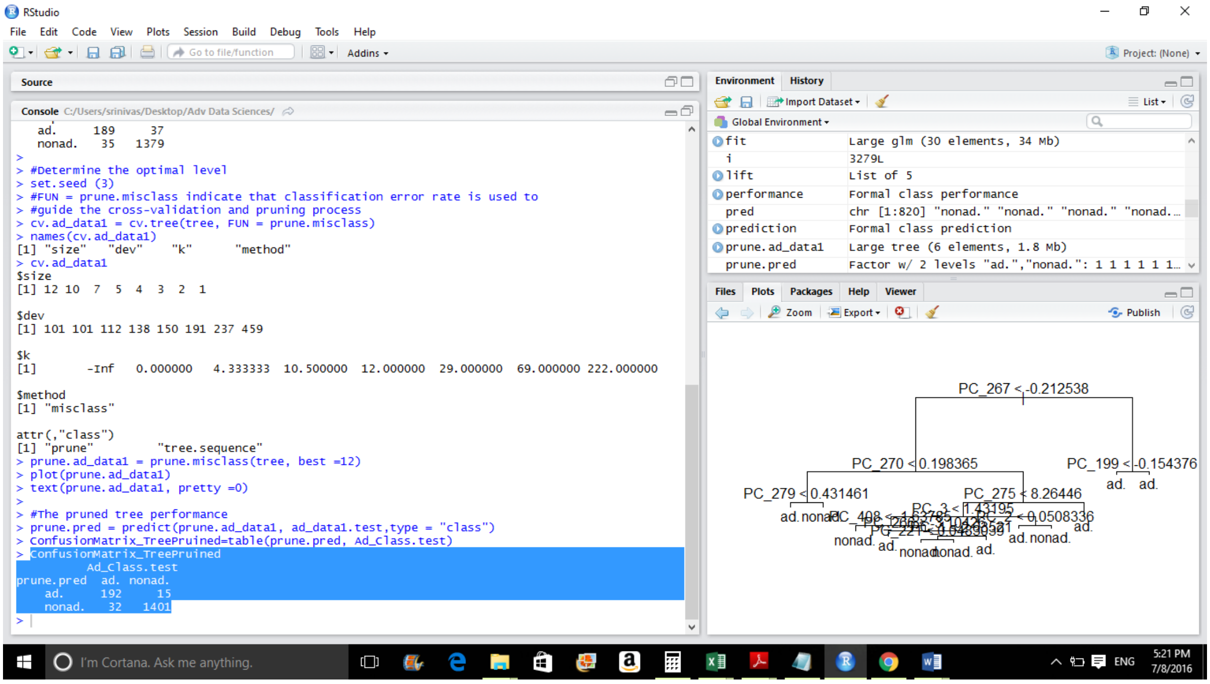
(b) Overall error rate_TreePruined=2.8%

NOTE: The overall error rate has come down after Pruning from 4.3% to 2.8%

Error rate and accuracy in prediciting each individual class

- ✓ Error rate in Predicting Ad class= 14/207= 7.2%
- ✓ Accuracy in Predicting Ad class= 187/207= 92.7%
- ✓ Error rate in Predicitng NonAD class= 32/1433= 2.2%
- ✓ Accuracy in Predicitng NonAD class= 1401/1433= 97.7%

This tree model which is not pruned is more accurate in predicting NonAD classes.

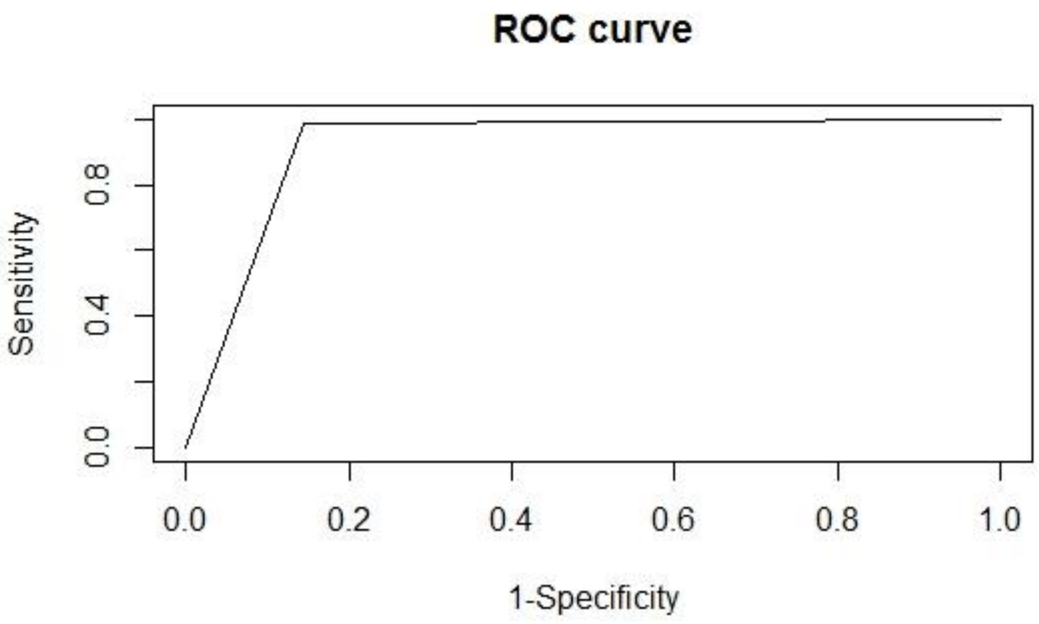


```
   ad.      189     37
   nonad.    35   1379
>
> #Determine the optimal level
> set.seed (3)
> #FUN = prune.misclass indicate that classification error rate is used to
> #guide the cross-validation and pruning process
> cv.ad_data1 = cv.tree(tree, FUN = prune.misclass)
> names(cv.ad_data1)
[1] "size"   "dev"    "k"      "method"
> cv.ad_data1
$size
[1] 12 10  7  5  4  3  2  1

$dev
[1] 101 101 112 138 150 191 237 459

$k
[1]       -Inf   0.000000   4.333333  10.500000  12.000000  29.000000  69.000000 222.000000

$method
[1] "misclass"

attr(,"class")
[1] "prune"         "tree.sequence"
> prune.ad_data1 = prune.misclass(tree, best =12)
> plot(prune.ad_data1)
> text(prune.ad_data1, pretty =0)
>
> #The pruned tree performance
> prune.pred = predict(prune.ad_data1, ad_data1.test,type = "class")
> ConfusionMatrix_TreePruined=table(prune.pred, Ad_Class.test)
> ConfusionMatrix_TreePruined
          Ad_Class.test
prune.pred  ad.  nonad.
    ad.     192     15
    nonad.   32   1401
>
```

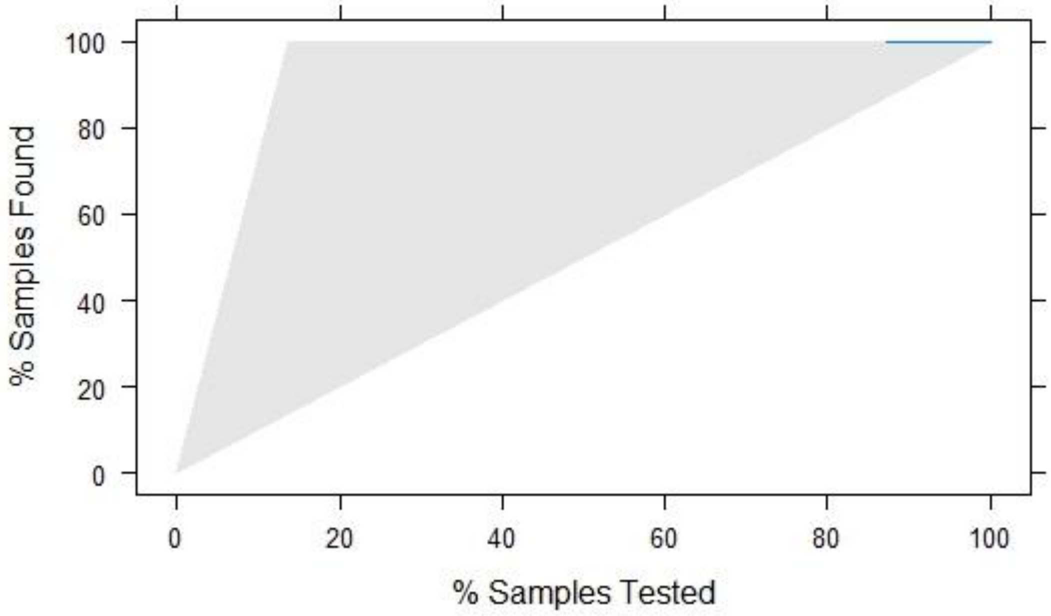
```
fit              Large glm (30 elements, 34 Mb)
i                3279L
lift             List of 5
performance      Formal class performance
pred             chr [1:820] "nonad." "nonad." "nonad." "nonad.…
prediction       Formal class prediction
prune.ad_data1   Large tree (6 elements, 1.8 Mb)
prune.pred       Factor w/ 2 levels "ad.","nonad.": 1 1 1 1 1 1…
```

In the above picture the highlighted blue part shows the confusion matrix of the tree model which is **pruned**.

<u>ROC CURVE FOR THE PRUNED TREE MODEL</u>



LIFT CHART FOR PRUNED TREE MODEL



Conclusion from classification tree model: Since the performance metrics and confusion matrix looks better after the tree is pruned we are considering this model which is pruned as our final classification tree model. **The error rate from this model is 2.8%**

LOGISTIC REGRESSION- CLASSIFICATION MODEL

Below are the confusion matrix and error rate of Logisitc Regression classification model

```
       Confusion Matrix and Statistics

          Reference
Prediction ad. Nonad.
    Ad.      23     94
    nonad. 677      26
```

Overall Error rate= ((677+94)/(820))

= 94%

Error rate and accuracy in prediciting each individual class

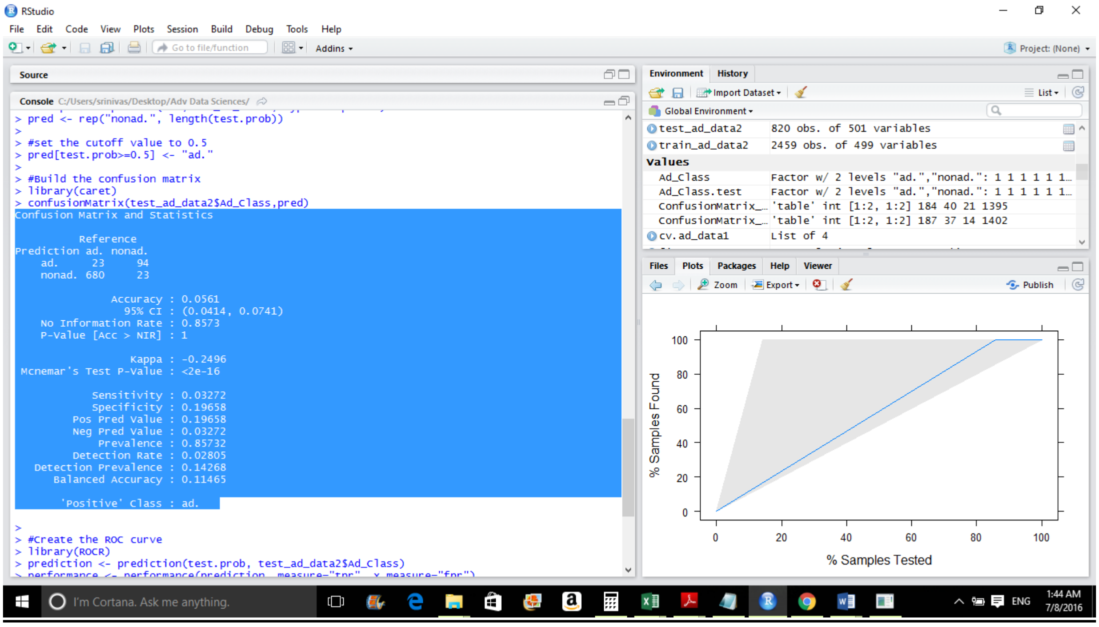
- ✓ Error rate in Predicting Ad class= 94/117= 80.3%
- ✓ Accuracy in Predicting Ad class= 23/117= 19.65%
- ✓ Error rate in Predicitng NonAD class= 680/703= 96.3%
- ✓ Accuracy in Predicitng NonAD class= 26/703= 3.6%

As we can see above the overall error rate is too high for the logistic regression model.

It predicts only 26AD cases correctly as ad and it predicts 94 AD cases wrongly as nonad

It Predicts only 26 NONAD cases correctly as nonad and it predicts 680 NONAD cases wrongly as ad

Since the accuracy rate of predicting AD class correctly is more than the accuracy of 8redicting NONAD class it is considered as a positive class for ad.

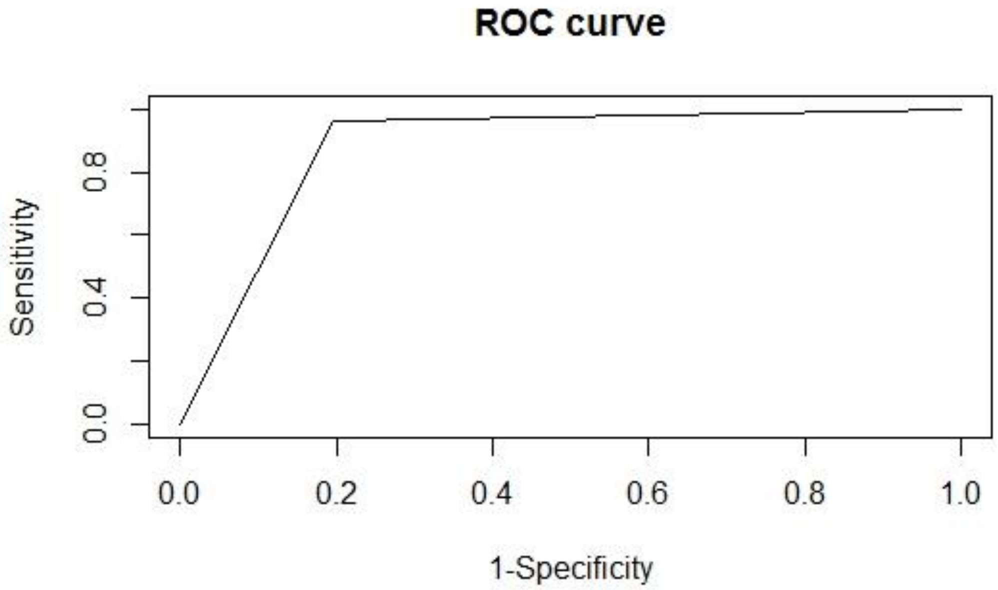


```
> pred <- rep("nonad.", length(test.prob))
>
> #set the cutoff value to 0.5
> pred[test.prob>=0.5] <- "ad."
>
> #Build the confusion matrix
> library(caret)
> confusionMatrix(test_ad_data2$Ad_Class,pred)
Confusion Matrix and Statistics

          Reference
Prediction ad. nonad.
    ad.     23     94
    nonad. 680     23

               Accuracy : 0.0561
                 95% CI : (0.0414, 0.0741)
    No Information Rate : 0.8573
    P-Value [Acc > NIR] : 1

                  Kappa : -0.2496
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.03272
            Specificity : 0.19658
         Pos Pred Value : 0.19658
         Neg Pred Value : 0.03272
             Prevalence : 0.85732
         Detection Rate : 0.02805
   Detection Prevalence : 0.14268
      Balanced Accuracy : 0.11465

       'Positive' Class : ad.

>
> #Create the ROC curve
> library(ROCR)
> prediction <- prediction(test.prob, test_ad_data2$Ad_Class)
> performance <- performance(prediction, measure="tpr", x.measure="fpr")
```

```
test_ad_data2        820 obs. of 501 variables
train_ad_data2       2459 obs. of 499 variables
Values
  Ad_Class           Factor w/ 2 levels "ad.","nonad.": 1 1 1 1 1 1…
  Ad_Class.test      Factor w/ 2 levels "ad.","nonad.": 1 1 1 1 1 1…
  ConfusionMatrix_…  'table' int [1:2, 1:2] 184 40 21 1395
  ConfusionMatrix_…  'table' int [1:2, 1:2] 187 37 14 1402
cv.ad_data1          List of 4
```

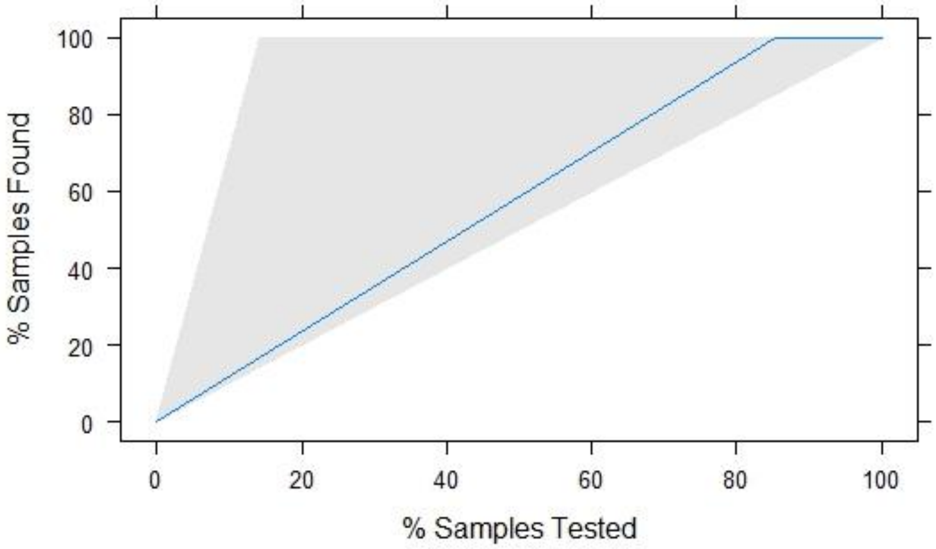
The Blue Highlighted part above shows the confusion matrix and different statistics of logistic regression.

ROC CURVE



The above mentioned picture shows the ROC curve for the logistic model built on Internet Advertisements dataset.

LIFE CHART



The above mentioned picture shows the Life Chart for the logistic regression classification model

NEURAL NETWORK-CLASSIFICATION MODEL

While fitting in the neural network model we should be careful in giving the vakues for size, weights, MaxNWts parameters. These values vary depending on the size of the predictors we are trying to fit. For this particular case where the predictors or variable size is huge we are MaxNWts=5001 and size as 5. The confusion Matrix for neural network is as follows

```
ConfusionMatrix_Neural_Network
        test.nnet
         ad. nonad.
  ad.    123     22
  nonad.   9    830
```

Overall Error Rate= ((22+9)/(984))*100

= 3.1

Error rate and accuracy in prediciting each individual class

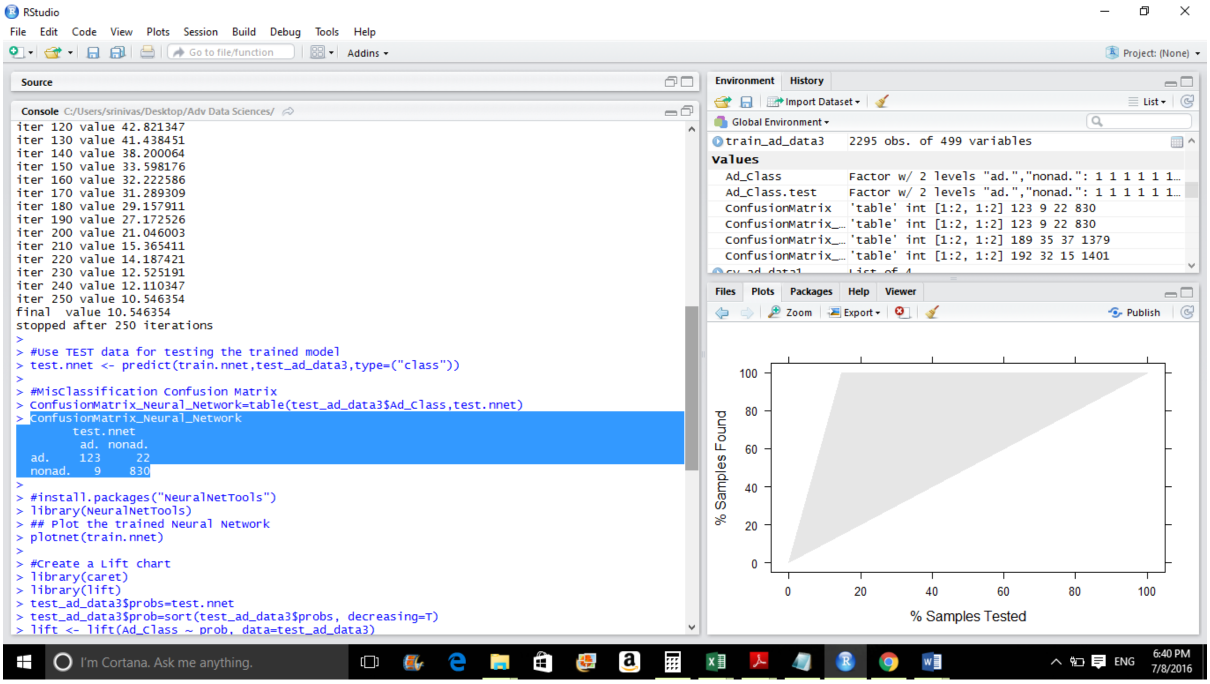
- ✓ Error rate in Predicting Ad class= 22/145= 15.1%
- ✓ Accuracy in Predicting Ad class= 123/145= 84.8%
- ✓ Error rate in Predicitng NonAD class= 9/839= 1%
- ✓ Accuracy in Predicitng NonAD class= 830/839= 98.9%

As we can see above the overall error rate is 3.1% for the neural network model.

It predicts only 123 AD cases correctly as ad and it predicts 22 AD cases wrongly as nonad

It Predicts only 830 NONAD cases correctly as nonad and it predicts 9 NONAD cases wrongly as ad

This model is more accurate in predicting the NonAD cases.

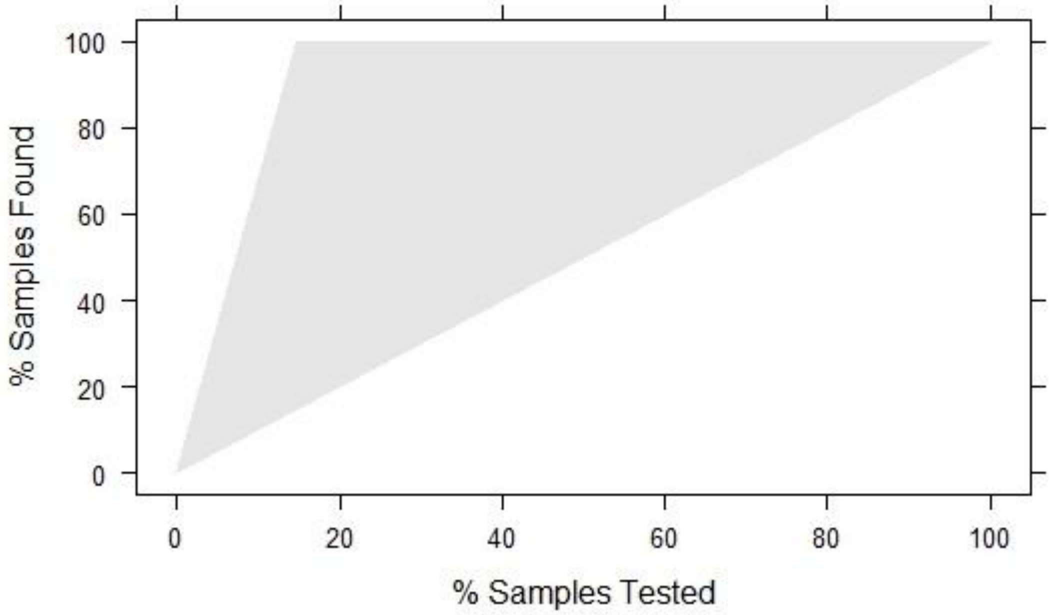


```
iter 120 value 42.821347
iter 130 value 41.438451
iter 140 value 38.200064
iter 150 value 33.598176
iter 160 value 32.222586
iter 170 value 31.289309
iter 180 value 29.157911
iter 190 value 27.172526
iter 200 value 21.046003
iter 210 value 15.365411
iter 220 value 14.187421
iter 230 value 12.525191
iter 240 value 12.110347
iter 250 value 10.546354
final  value 10.546354
stopped after 250 iterations
>
> #Use TEST data for testing the trained model
> test.nnet <- predict(train.nnet,test_ad_data3,type=("class"))
>
> #MisClassification Confusion Matrix
> ConfusionMatrix_Neural_Network=table(test_ad_data3$Ad_Class,test.nnet)
> ConfusionMatrix_Neural_Network
        test.nnet
         ad. nonad.
  ad.    123     22
  nonad.   9    830
>
> #install.packages("NeuralNetTools")
> library(NeuralNetTools)
> ## Plot the trained Neural Network
> plotnet(train.nnet)
>
> #Create a Lift chart
> library(caret)
> library(lift)
> test_ad_data3$probs=test.nnet
> test_ad_data3$prob=sort(test_ad_data3$probs, decreasing=T)
> lift <- lift(Ad_Class ~ prob, data=test_ad_data3)
```

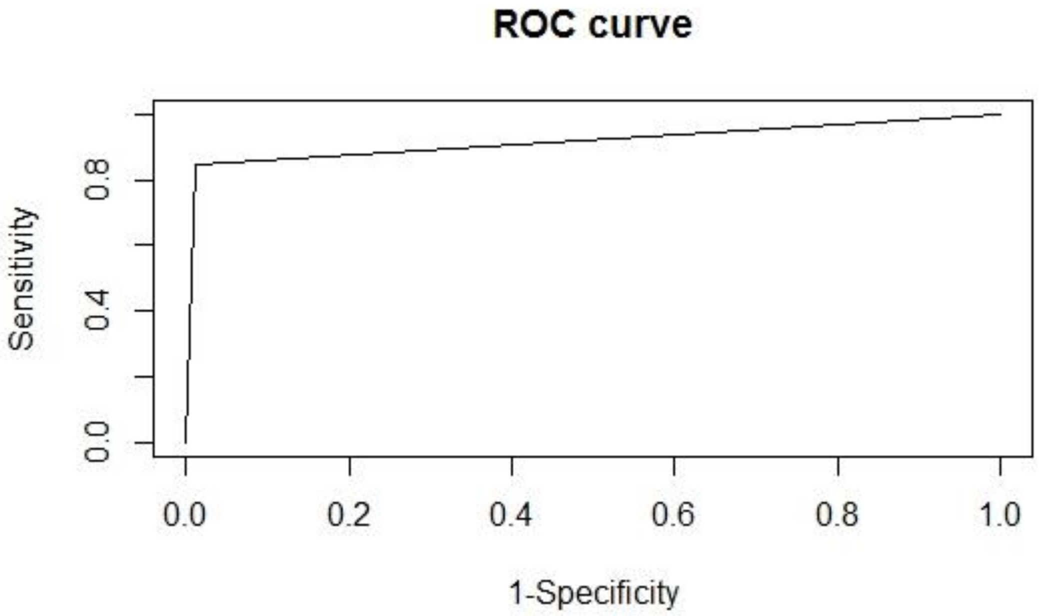
```
train_ad_data3      2295 obs. of 499 variables
Values
  Ad_Class          Factor w/ 2 levels "ad.","nonad.": 1 1 1 1 1 1…
  Ad_Class.test     Factor w/ 2 levels "ad.","nonad.": 1 1 1 1 1 1…
  ConfusionMatrix   'table' int [1:2, 1:2] 123 9 22 830
  ConfusionMatrix_… 'table' int [1:2, 1:2] 123 9 22 830
  ConfusionMatrix_… 'table' int [1:2, 1:2] 189 35 37 1379
  ConfusionMatrix_… 'table' int [1:2, 1:2] 192 32 15 1401
```

The above highlighted part shows the confusion matrix of neural network model

LIFT CHART NEURAL NETWORK



ROC CURVE FOR NEURAL NETWORK



FINAL CONCLUSION FOR PROBLEM 2 which model we would choose

After comparing the performance metrics of all the three classification models for the dataset Internet Advertisements-Problem 2 we decided that we would choose **Classification trees** model. We have decided this because the overall error rate is less in that model compared to other two models. Also the accuracy and error rates in predicting the individual classes: Ad, NonAd is better in **Classification trees** model than the other two models.

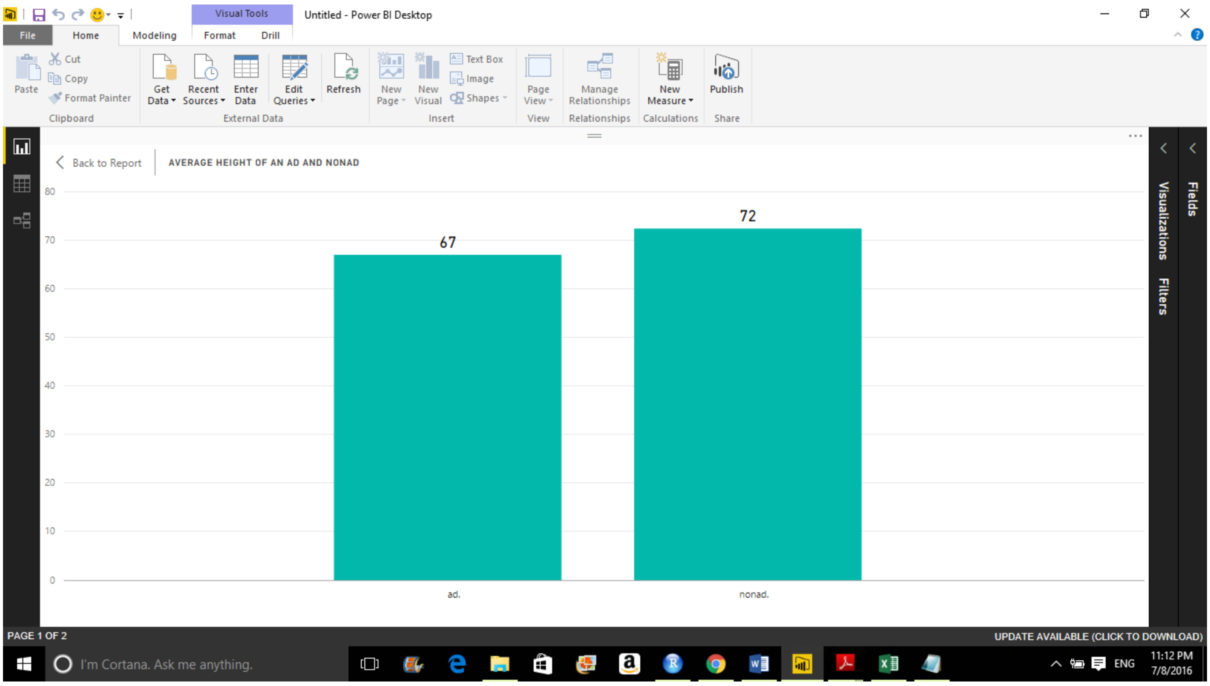
REFERENCES

1. http://www.fileformat.info/tip/web/imagesize.html
2. **http://www.r-bloggers.com/computing-and-visualizing-pca-in-r/**
3. **http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/lecturas-clasificacion/abstracts-resumir/kushmerick99learning.pdf**

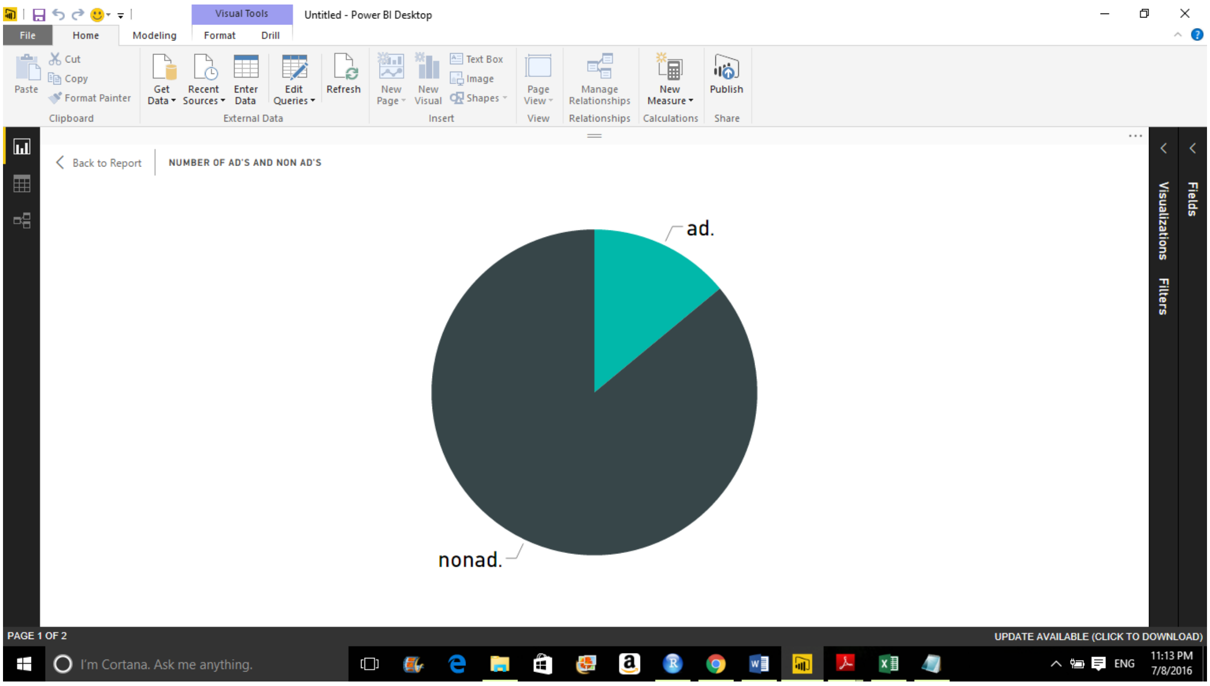
POWER BI VISUALIZATIONS

These Visualizations give information regarding the geometrical aspects of Internet Advertisements such as Height, Width, Aspect Ratio. It also gives information regarding number of Ad's deployed in local and non-local servers.

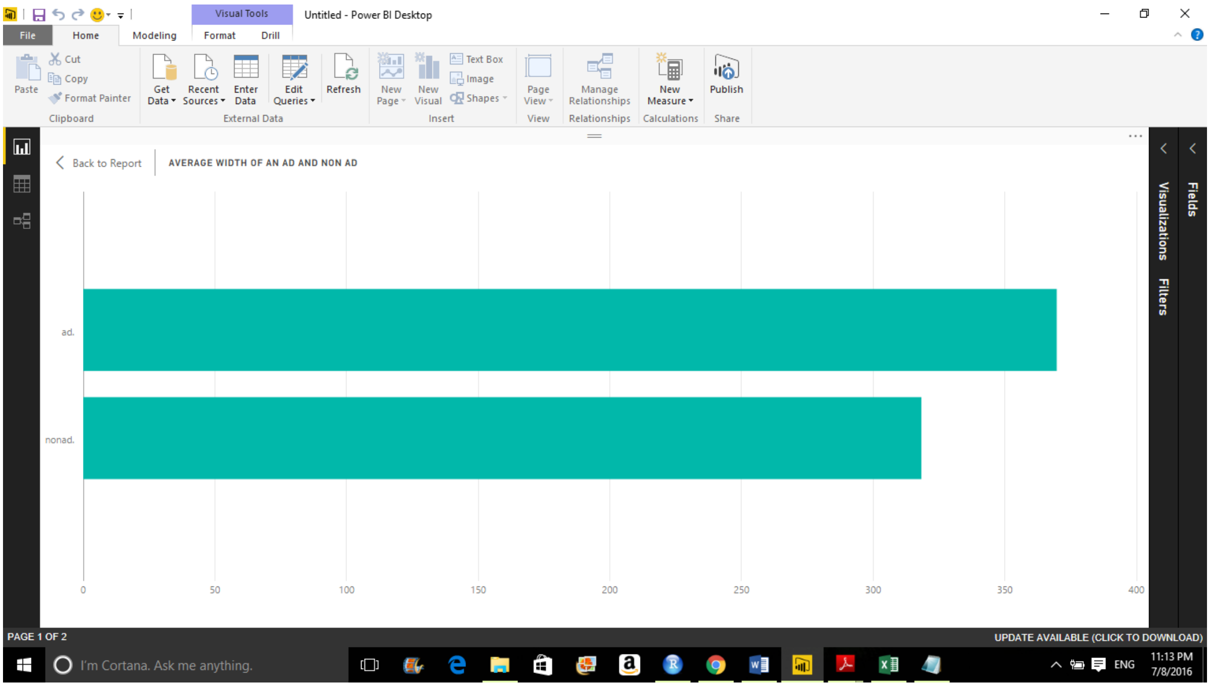
Note: These Visualizations were done after replacing the missing values in continuos data i.e in Height, width and aspect ratio features.



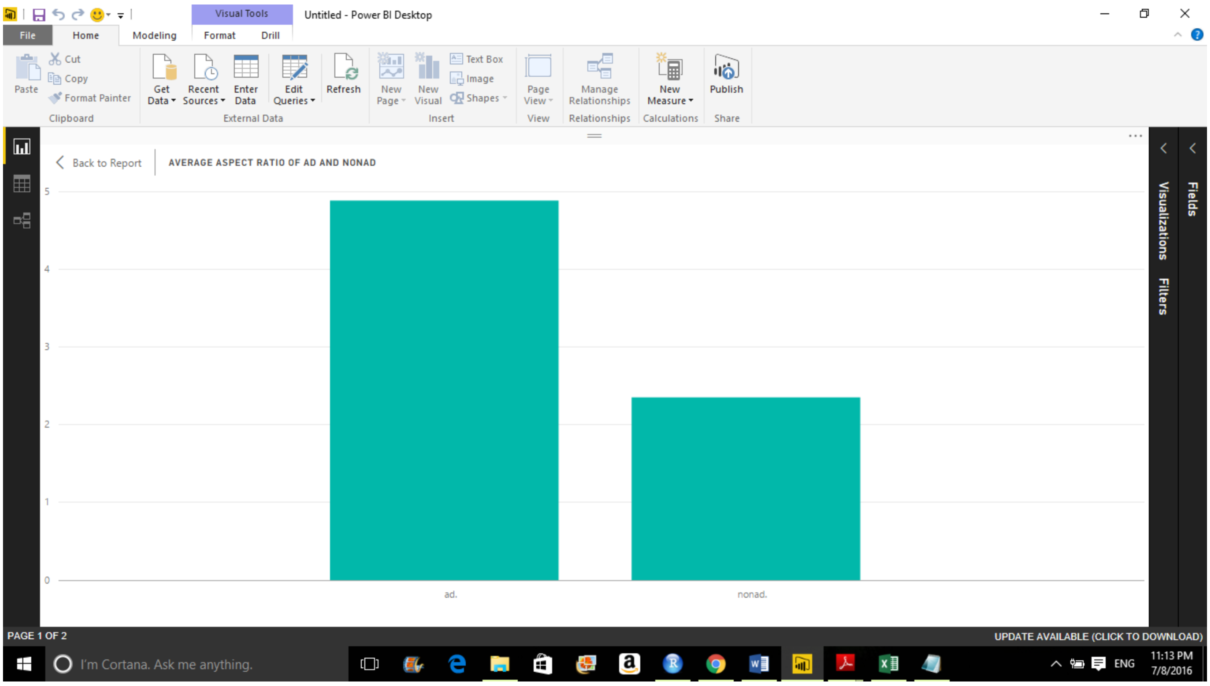
The above Visualization shows the average height of an Ad and Non Ad reported in the dataset


Untitled - Power BI Desktop
Visual Tools
File
Home
Modeling
Format
Drill
Back to Report
NUMBER OF AD'S AND NON AD'S
ad.
nonad.
Visualizations
Filters
Fields
PAGE 1 OF 2
UPDATE AVAILABLE (CLICK TO DOWNLOAD)
I'm Cortana. Ask me anything.
11:13 PM
7/8/2016

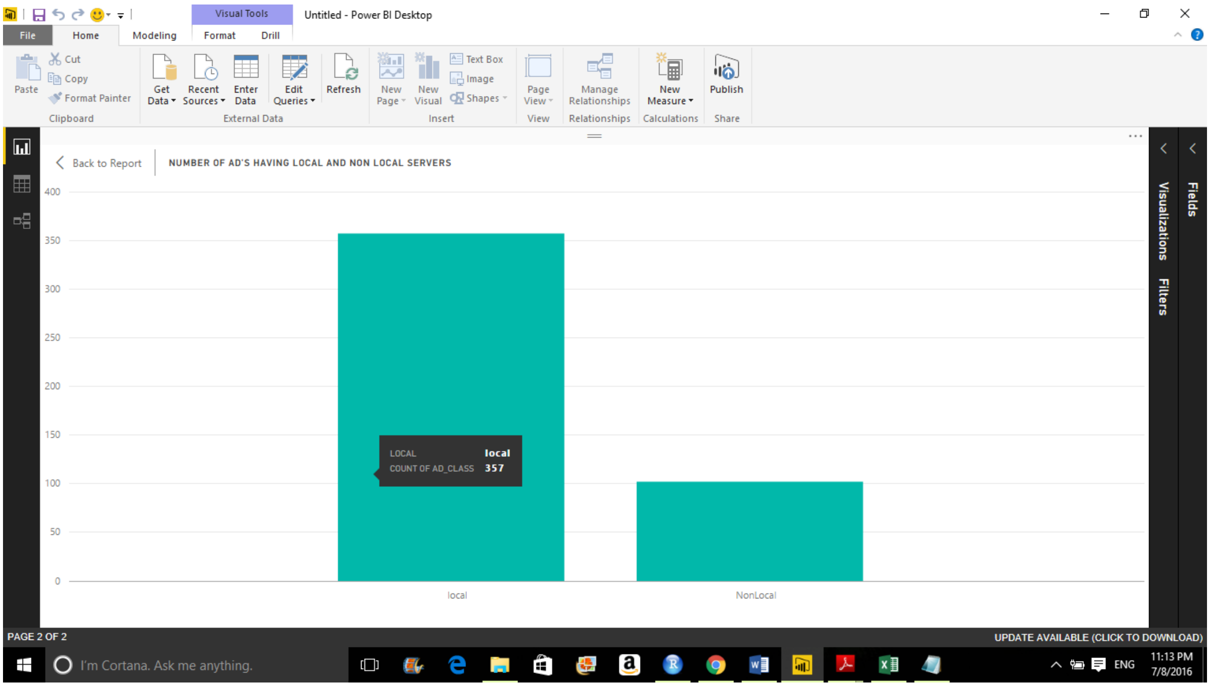
The above visualization shows the number of AD’s and NonAd’s



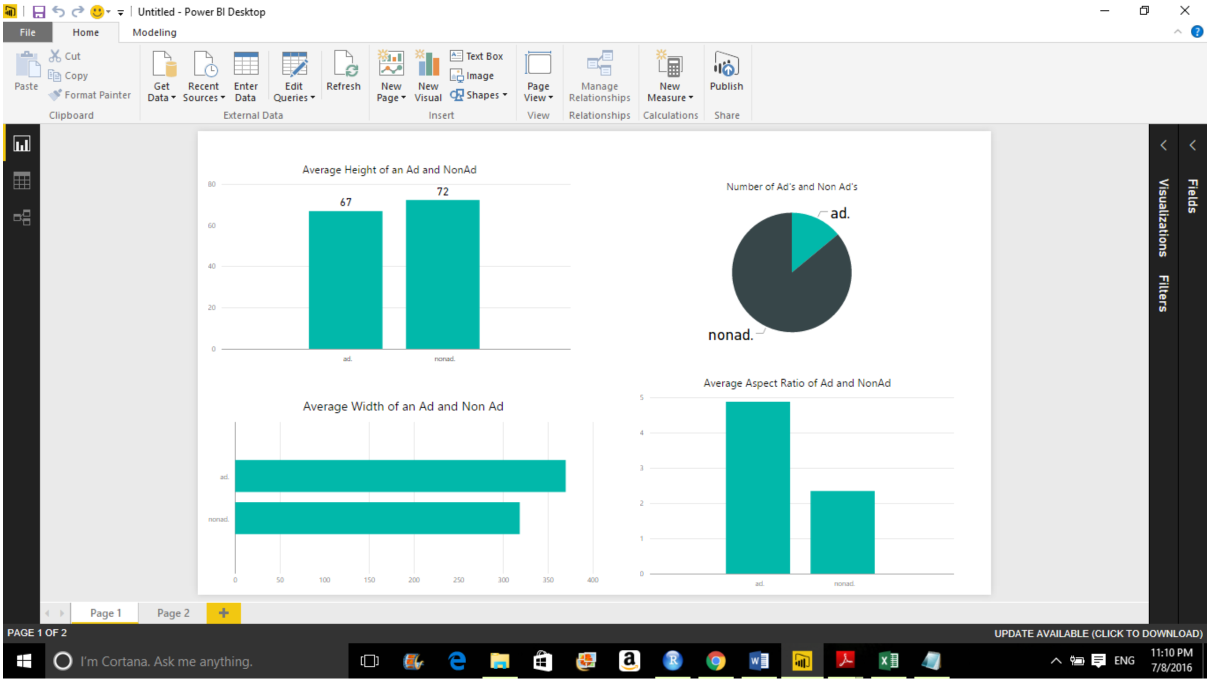
The above visualization shows the Average Width of Ad’s and Non Ad’s



The above visualization shows the average aspect ratio of Ad’s and Non AD’s



The above visualization shows the number of Ad’s which are in local and non-local servers



The above picture is a dashboard of all the visualizations together