`

CONFIGURATION MANUAL

Use of Machine Learning Techniques for Integrating Source Data from

MSc Research Project
Data Analytics

Shanthi Marie Teres Alexis

School of Computing
National College of Ireland

Supervisor: Dr. Paul Hayes

Table of Contents

INTRODUCTION

This manual acts as a guide to the setup and steps run in using machine learning techniques in integrating source data to destination. The steps provided are not a full analysis but a manual on the basic initial steps and executed algorithms. As there are many options and parameters that can be applied on algorithms that will vary the result produced; self-initiatives to explore other parameter inputs is encouraged.

SYSTEM CONFIGURATION

Machine requirement used is as below. Although the requirement is not mandatory, it is advisable to have sufficient memory in place as the algorithms run is resource hungry.

Windows edition

Windows 10 Home Single Language

System

Processor:	Intel(R) Core(TM) i7-4710HQ CPU @ 2.50GHz 2.50 GHz
Installed memory (RAM):	16.0 GB
System type:	64-bit Operating System, x64-based processor

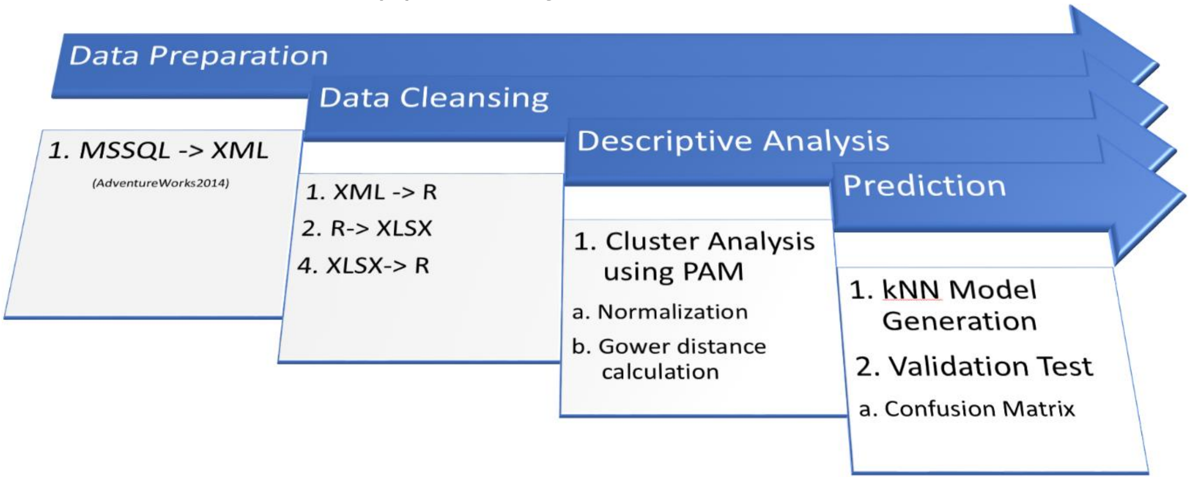
APPLICATION REQUIREMENT

Application used for this analysis are:

Process	Application	Setup Required	Notes
Data Preparation	MS SQL Server 2014	AdventureWorks2014 database	Downloadable at : http://msftdbprodsamples.codeplex.com/releases/view/55330
		AdventureWorksDW2014 datawarehouse	
	MS Excel 2013	Any standard distribution type is sufficient	
Data Cleansing and Analysis	Microsoft R Open 3.2.5		

WORKFLOW DIAGRAM

The overall flow of this research project analysis is as below which consists of a data preparation phase, data cleansing, descriptive analysis and the prediction model generation and evaluation phase. The following sections will detail the actual scripts and results derived and any processing that was executed.



DATA PREPARATION

The objective of this phase is to create a XML file as would be expected in the real-world scenario. Preliminary steps of downloading and importing the AdventureWorks2014 database and data warehouse need to be completed before continuing with this steps. The documentation and setup files are available at this CodePlex[1] site.

Once the database and data warehouse are setup, navigate to the AdventureWorks2014 database. The following script is executed to extract the XML format for further processing.

[1] http://msftdbprodsamples.codeplex.com/releases/view/55330

Extraction Script:

```
drop table TMP_DISS_TBL;

SELECT    p.[BusinessEntityID]
     ,p.[FirstName]
     ,p.[MiddleName]
     ,p.[LastName]
     ,p.NameStyle
 ,CONVERT(datetime, REPLACE([IndividualSurvey].[ref].[value](N'declare default element namespace "
 http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/IndividualSurvey";
BirthDate[1]', 'nvarchar(20)') ,'Z', ''), 101) AS [BirthDate]
,[IndividualSurvey].[ref].[value](N'declare default element namespace "
http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/IndividualSurvey";
MaritalStatus[1]', 'nvarchar(1)') AS [MaritalStatus]
,[IndividualSurvey].[ref].[value](N'declare default element namespace "
http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/IndividualSurvey";
Gender[1]', 'nvarchar(1)') AS [Gender]
     ,p.[ModifiedDate] into TMP_DISS_TBL
FROM [Person].[Person] as p
OUTER APPLY [AdditionalContactInfo].nodes(
    'declare namespace ci="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ContactInfo";
    /ci:AdditionalContactInfo') AS ContactInfo(ref)
cross APPLY [Demographics].nodes(
    'declare namespace ci="
    http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/IndividualSurvey";
    /ci:IndividualSurvey') AS IndividualSurvey(ref)
  join person.EmailAddress as ea on p.businessEntityID = ea.businessentityID
  join person.PersonPhone as pphone on p.businessEntityID = pphone.businessentityID

select * from TMP_DISS_TBL
where BirthDate is not null
and MaritalStatus is not null
and gender is not null
for XML AUTO, ELEMENTS, XMLSCHEMA('person');
```

XML Document:


```
</xsd:restriction></xsd:simpleType></xsd:element></xsd:sequence></xsd:complexType></xsd:element></xsd:sc
hema><p xmlns="person"><BusinessEntityID>1708</BusinessEntityID><FirstName>Linda
</FirstName><MiddleName>R.</MiddleName><LastName>Rousey</LastName><NameStyle>0</NameStyle><BirthDate>
04/07/1947</BirthDate><MaritalStatus>M</MaritalStatus><Gender>F</Gender><ModifiedDate>04/07/1947
</ModifiedDate></p><p xmlns="person"><BusinessEntityID>1715</BusinessEntityID><FirstName>Justine
</FirstName><MiddleName>J.</MiddleName><LastName>Ryan</LastName><NameStyle>0</NameStyle><BirthDate>
11/11/1947</BirthDate><MaritalStatus>S</MaritalStatus><Gender>F</Gender><ModifiedDate>11/11/1947
</ModifiedDate></p><p xmlns="person"><BusinessEntityID>1722</BusinessEntityID><FirstName>Mandar
</FirstName><LastName>Samant</LastName><NameStyle>0</NameStyle><BirthDate>09/18/1974
</BirthDate><MaritalStatus>S</MaritalStatus><Gender>F</Gender><ModifiedDate>09/18/1974
</ModifiedDate></p><p xmlns="person"><BusinessEntityID>1733</BusinessEntityID><FirstName>K.
</FirstName><LastName>Saravan</LastName><NameStyle>0</NameStyle><BirthDate>07/02/1979
</BirthDate><MaritalStatus>S</MaritalStatus><Gender>M</Gender><ModifiedDate>07/02/1979
</ModifiedDate></p><p xmlns="person"><BusinessEntityID>1740</BusinessEntityID><FirstName>Scott
</FirstName><LastName>Seely
</LastName><NameStyle>0</NameStyle><BirthDate>11/19/1947</BirthDate><MaritalStatus>M
</MaritalStatus><Gender>F</Gender><ModifiedDate>11/19/1947</ModifiedDate></p><p xmlns="person"
><BusinessEntityID>1747</BusinessEntityID><FirstName>David</FirstName><MiddleName>B.
</MiddleName><LastName>Shepard</LastName><NameStyle>0</NameStyle><BirthDate>11/21/1957
</BirthDate><MaritalStatus>M</MaritalStatus><Gender>F</Gender><ModifiedDate>11/21/1957
</ModifiedDate></p><p xmlns="person"><BusinessEntityID>1758</BusinessEntityID><FirstName>Anne
</FirstName><MiddleName>R.</MiddleName><LastName>Sims</LastName><NameStyle>0</NameStyle><BirthDate>
08/05/1963</BirthDate><MaritalStatus>M</MaritalStatus><Gender>M</Gender><ModifiedDate>08/05/1963
</ModifiedDate></p><p xmlns="person"><BusinessEntityID>1765</BusinessEntityID><FirstName>Cathy
</FirstName><MiddleName>J.</MiddleName><LastName>Sloan</LastName><NameStyle>0</NameStyle><BirthDate>
07/26/1953</BirthDate><MaritalStatus>M</MaritalStatus><Gender>M</Gender><ModifiedDate>07/26/1953
</ModifiedDate></p><p xmlns="person"><BusinessEntityID>1772</BusinessEntityID><FirstName>Jeff
</FirstName><LastName>Smith</LastName><NameStyle>0</NameStyle><BirthDate>07/10/1949
</BirthDate><MaritalStatus>M</MaritalStatus><Gender>M</Gender><ModifiedDate>07/10/1949
</ModifiedDate></p><p xmlns="person"><BusinessEntityID>1783</BusinessEntityID><FirstName>Sylvia
</FirstName><MiddleName>N.</MiddleName><LastName>Spencer</LastName><NameStyle>0</NameStyle><BirthDate>
10/11/1951</BirthDate><MaritalStatus>M</MaritalStatus><Gender>M</Gender><ModifiedDate>10/11/1951
</ModifiedDate></p><p xmlns="person"><BusinessEntityID>2383</BusinessEntityID><FirstName>Crystal
</FirstName><MiddleName>L</MiddleName><LastName>Cai</LastName><NameStyle>0</NameStyle><BirthDate>
08/08/1966</BirthDate><MaritalStatus>M</MaritalStatus><Gender>F</Gender><ModifiedDate>08/08/1966
</ModifiedDate></p><p xmlns="person"><BusinessEntityID>2390</BusinessEntityID><FirstName>Laura
</FirstName><MiddleName>P</MiddleName><LastName>Huang</LastName><NameStyle>0</NameS
tyle><BirthDate>11/06/1947</BirthDate><MaritalStatus>M</MaritalStatus><Gender>F</Gender><ModifiedDate>
11/06/1947</ModifiedDate></p><p xmlns="person"><BusinessEntityID>2394</BusinessEntityID><FirstName>
```


Once the XML document is generated it should be saved into a local folder for the data cleansing phase.

DATA CLEANSING

The data cleansing will involve these steps:

- Importing the XML through R to be parsed and broken into its element details.
- Cleansing of parsed elements in XML and data type determination.
  Ideally this step will not exist as the cleansing will be automatically executed R using a function written in C#. Unfortunately, the function was not completed within the expected timeline and an interim work-around solution was created using Excel.
- Import into R for analysis

Importing and Parsing XML through R

This steps requires the RVEST and XML packages to be installed.
The Script:

```
### 1. Load to parse the data
## install.packages("xml2")
## install.packages("rvest")
library(xml2)
library(rvest)
page<-read_html("C:\\dissertation\\scripts\\02b_ext_in_xml.xml")
persons<-html_nodes(page,  xpath = "//p")
fieldnames<-xml_name(xml_find_all(persons, ".//*"))
fields<-xml_text(xml_find_all(persons, ".//*"))
df<-data.frame(fieldnames, fields)
```

The Result:

	fieldnames	fields
1	businessentityid	1708
2	firstname	Linda
3	middlename	R.
4	lastname	Rousey
5	namestyle	0
6	birthdate	04/07/1947
7	maritalstatus	M
8	gender	F
9	modifieddate	04/07/1947
10	businessentityid	1715
11	firstname	Justine
12	middlename	J.
13	lastname	Ryan
14	namestyle	0
15	birthdate	11/11/1947

Cleansing of Parsed Elements

This steps requires the parsed elements to be imported from R into Excel where some formulas are run to extract the cleaned data. Since R cannot be imported directly to Excel, a CSV file is exported.

```
### 02.Export parsed data for cleasning in Excel
setwd("C:/dissertation/scripts")
write.csv(df, "C:/dissertation/script/04b_xmlData.csv")
```

	A	B	C	D	E
1	No	fieldnames	fields	data type	mapped_flg
2	1	businessentityid	1708	number	UNK
3	2	firstname	Linda	character	DIM_CUSTOMER/FirstName
4	3	middlename	R.	character	DIM_CUSTOMER/MiddleNAme
5	4	lastname	Rousey	character	DIM_CUSTOMER/LastName
6	5	namestyle	0	number	DIM_CUSTOMER/NameStyle
7	6	birthdate	04/07/1947	date	DIM_CUSTOMER/BirthDate
8	7	maritalstatus	M	character	DIM_CUSTOMER/MaritalStatus
9	8	gender	F	character	DIM_CUSTOMER/Gender
10	9	modifieddate	04/07/1947	date	UNK
11	10	businessentityid	1715	number	UNK
12	11	firstname	Justine	character	DIM_CUSTOMER/FirstName
13	12	middlename	J.	character	DIM_CUSTOMER/MiddleNAme
14	13	lastname	Ryan	character	DIM_CUSTOMER/LastName
15	14	namestyle	0	number	DIM_CUSTOMER/NameStyle
16	15	birthdate	11/11/1947	date	DIM_CUSTOMER/BirthDate
17	16	maritalstatus	S	character	DIM_CUSTOMER/MaritalStatus
18	17	gender	F	character	DIM_CUSTOMER/Gender
19	18	modifieddate	11/11/1947	date	UNK
20	19	businessentityid	1722	number	UNK
21	20	firstname	Mandar	character	DIM_CUSTOMER/FirstName
22	21	lastname	Samant	character	DIM_CUSTOMER/LastName
23	22	namestyle	0	number	DIM_CUSTOMER/NameStyle
24	23	birthdate	09/18/1974	date	DIM_CUSTOMER/BirthDate
25	24	maritalstatus	S	character	DIM_CUSTOMER/MaritalStatus
26	25	gender	F	character	DIM_CUSTOMER/Gender
27	26	modifieddate	09/18/1974	date	UNK
28	27	businessentityid	1733	number	UNK
29	28	firstname	K.	character	DIM_CUSTOMER/FirstName
30	29	lastname	Saravan	character	DIM_CUSTOMER/LastName
31	30	namestyle	0	number	DIM_CUSTOMER/NameStyle

Next, the CSV file is imported into Excel with the formulas applied to column E and D detailed below.

Formula Column D:

```
=IF(B2="yearlyincome","number",
IF(B2="totalpurchaseytd","number",
IF(B2="totalchildren","number",
IF(B2="rowguid","raw",
IF(B2="persontype","number",
IF(B2="occupation","character",
IF(B2="numberchildrenathome","number",
IF(B2="numbercarsowned","number",
IF(B2="namestyle","number",
IF(B2="middlename","character",
IF(B2="lastname","character",
IF(B2="individualsurvey","number",
IF(B2="homeownerflag","number",
IF(B2="gender","character",
IF(B2="firstname","character",
IF(B2="emailpromotion","number",
IF(B2="datefirstpurchase","date",
IF(B2="commutedistance","number",
IF(B2="businessentityid","number",
IF(B2="birthdate","date",
IF(B2="modifieddate","date",
IF(B2="maritalstatus","character",
IF(B2="education","character",
IF(B2="street","character",
IF(B2="city","character",
IF(B2="stateprovince","character",
IF(B2="postalcode","character",
IF(B2="countryregion","character",
IF(B2="rowguid","guid",
IF(B2="modifieddate","date",
IF(B2="title","character",
IF(B2="suffix","character","")))))))))))))))))))))))))))))))))
```

Description: To identify the character type of the data based on the element content. Initial plan was to do this via a C# which still under progress. The use of C# as is it flexible, a native to most computer and hence would computer faster in R.

Formula Column E:

```
=IF(B2="namestyle","DIM_CUSTOMER/NameStyle",
IF(B2="occupation","DIM_CUSTOMER/EnglishOcupation",
IF(B2="suffix","DIM_CUSTOMER/Suffix",
IF(B2="title","DIM_CUSTOMER/Title",
IF(B2="firstname","DIM_CUSTOMER/FirstName",
IF(B2="middlename","DIM_CUSTOMER/MiddleNAme",
IF(B2="lastname","DIM_CUSTOMER/LastName",
IF(B2="birthdate","DIM_CUSTOMER/BirthDate",
IF(B2="gender","DIM_CUSTOMER/Gender",
IF(B2="stateprovince","DIM_GEOGRAPHY/StateProvinceName",
IF(B2="street","DIM_CUSTOMER/AddressLine1",
IF(B2="city","DIM_GEOGRAPHY/City",
IF(B2="countryregion","DIM_GEOGRAPHY/EnglishCountryRegionName",
```

```
IF(B2="postalcode","DIM_GEOGRAPHY/PostalCode",
IF(B2="datefirstpurchase","DIM_CUSTOMER/DateFIrstPurchased",
IF(B2="homeownerflag","DIM_CUSTOMER/HouseOwnerFlag",
IF(B2="namestyle","DIM_CUSTOMER/NameStyle",
IF(B2="numbercarsowned","DIM_CUSTOMER/NumberCarsOwned",
IF(B2="numberchildrenathome","DIM_CUSTOMER/NumberChildrenAtHome",
IF(B2="totalchildren","DIM_CUSTOMER/TotalChildren",
IF(B2="yearlyincome","DIM_CUSTOMER/YearlyINcome",
IF(B2="education","DIM_CUSTOMER/EnglishEducation",
IF(B2="businessentityid","UNK",
IF(B2="rowguid","UNK",IF(B2="modifieddate","UNK",
IF(B2="maritalstatus","DIM_CUSTOMER/MaritalStatus","")))))))))))))))))))))))))
```

Description: To create the class field for the supervised learning. This information is retrieved from the SSIS process that integrated the Database to the data warehouse provided by Microsoft. For systems where this information is unknown, the mapping will be determined by a domain expert.

During the R parsing process, the script concatenates the XML into a long string and breaks it each time it finds an element tag i.e. symbolized with <> until it finds the closing tag i.e. </>. In this process, it will also break the concatenated string as seen below which is cleaned manually for now. This processing can be automated in the C# function.

1	No	fieldnames	fields	data typ	mapped_flg
74	474	busines	2751NatalieGonzales011/09/1954MF11/09/1954		
21	722	businessentit	2969ErickWMadan005/03/1975SM05/03/1975		
46	847	namesty	008/25/1944SF08/25/1944		
70	971	busines	3212HaileyPRussell003/26/1978SF03/26/1978		

Cleansed data that character type is not identified should be written as an error file as this information is required by the machine learning algorithm to identify the target destination.
Subsequent from this step, the finalized cleaned data is imported into R as seen below:

The Script:

```
### 6. Process Cleaned XMl
dfCleanedDat<- read.csv("C:/dissertation/script/06_xmlData_cleaned2.csv", header = TRUE)


### 7.Get basic idea of data
View(dfCleanedDat)
dim(dfCleanedDat)
summary(dfCleanedDat)
str(dfCleanedDat)
levels(dfCleanedDat$mapped_flg)
```

The Result :

```
> ### 6. Process Cleaned XMl
> dfCleanedDat<- read.csv("C:/dissertation/script/06_xmlData_cleaned2.csv", head$
>
>
> ### 7.Get basic idea of data
> View(dfCleanedDat)
> dim(dfCleanedDat)
[1] 156858      4
> summary(dfCleanedDat)
         fieldnames          fields          data.type
 gender       :18369   M      :20253   character:83794
 modifieddate :18367   0      :18290   date     :36572
 lastname     :18341   F      : 9284   number   :36492
 firstname    :18313   S      : 8837
 namestyle    :18294   A      : 1281
 maritalstatus:18227   L      : 1241
 (Other)      :46947   (Other):97672
                      mapped_flg
 UNK                       :36565
 DIM_CUSTOMER_Gender       :18369
 DIM_CUSTOMER_LastName     :18341
 DIM_CUSTOMER_FirstName    :18313
 DIM_CUSTOMER_NameStyle    :18294
 DIM_CUSTOMER_MaritalStatus:18227
 (Other)                   :28749
> str(dfCleanedDat)
'data.frame':   156858 obs. of  4 variables:
 $ fieldnames: Factor w/ 9 levels "birthdate","businessentityid",..: 2 3 7 5 9 1$
 $ fields    : Factor w/ 27609 levels "","0","01-Apr",..: 15277 27071 27266 2733$
 $ data.type : Factor w/ 3 levels "character","date",..: 3 1 1 1 3 2 1 1 2 3 ...
 $ mapped_flg: Factor w/ 8 levels "DIM_CUSTOMER_BirthDate",..: 8 2 6 4 7 1 5 3 8$
> levels(dfCleanedDat$mapped_flg)
[1] "DIM_CUSTOMER_BirthDate"     "DIM_CUSTOMER_FirstName"
[3] "DIM_CUSTOMER_Gender"        "DIM_CUSTOMER_LastName"
[5] "DIM_CUSTOMER_MaritalStatus" "DIM_CUSTOMER_MiddleNAme"
[7] "DIM_CUSTOMER_NameStyle"     "UNK"
>
```

	fieldnames	fields	data.type	mapped_flg
1	businessentityid	1708	number	UNK
2	firstname	Linda	character	DIM_CUSTOMER_FirstName
3	middlename	R.	character	DIM_CUSTOMER_MiddleNAme
4	lastname	Rousey	character	DIM_CUSTOMER_LastName
5	namestyle	0	number	DIM_CUSTOMER_NameStyle
6	birthdate	04/07/1947	date	DIM_CUSTOMER_BirthDate
7	maritalstatus	M	character	DIM_CUSTOMER_MaritalStatus
8	gender	F	character	DIM_CUSTOMER_Gender
9	modifieddate	04/07/1947	date	UNK
10	businessentityid	1715	number	UNK
11	firstname	Justine	character	DIM_CUSTOMER_FirstName
12	middlename	J.	character	DIM_CUSTOMER_MiddleNAme
13	lastname	Ryan	character	DIM_CUSTOMER_LastName
14	namestyle	0	number	DIM_CUSTOMER_NameStyle
15	birthdate	11/11/1947	date	DIM_CUSTOMER_BirthDate
16	maritalstatus	S	character	DIM_CUSTOMER_MaritalStatus
17	gender	F	character	DIM_CUSTOMER_Gender
18	modifieddate	11/11/1947	date	UNK
19	businessentityid	1722	number	UNK
20	firstname	Mandar	character	DIM_CUSTOMER_FirstName
21	lastname	Samant	character	DIM_CUSTOMER_LastName
22	namestyle	0	number	DIM_CUSTOMER_NameStyle

Next, the profile information of elements and element contents need to be concatenated to the data frame.

The Script:

```
### 8.Make a a working data frame copy i.e. : mydata1 of unique information
mydata1<-dfCleanedDat
require(plyr)
mydata1Pred_grp <- unique(mydata1 [,c("fieldnames" , "mapped_flg")])

### 9. Add lengths of the element name & element values
mydata1$field_len <- nchar(as.character(mydata1$fields))
mydata1$fieldnames_len <- nchar(as.character(mydata1$fieldnames))
View(mydata1)

### 10. Where no value was available in column, assign 0 to the newly calculated length columns
###  mutate depend on dplyr
###  stringr depends on stringr
mydata1 <- mydata1%>%
   mutate(field_len = ifelse(is.na(field_len),0,field_len))
print
mydata1 <- mydata1%>%
   mutate(fieldnames_len= ifelse(is.na(fieldnames_len),0 ,fieldnames_len))

### install.packages("aggregate")
require(aggregate)
library(fitdistrplus)

### 11. Based on the element name length and element value length information - get a grouped by infor
###     for average length, max length and min length

distr.mean <- data.frame(aggregate(x = mydata1[c("field_len",
                                   "fieldnames_len")],
                            by = list(fieldnames = mydata1$fieldnames,field_datatype = mydata1$data.type),
                            mean))

distr.max <- data.frame(aggregate(x = mydata1[c("field_len",
                                  "fieldnames_len")],
                            by = list(fieldnames= mydata1$fieldnames,field_datatype = mydata1$data.type),

                            max))

distr.min <- data.frame(aggregate(x = mydata1[c("field_len",
                                  "fieldnames_len")],
                            by = list(fieldnames= mydata1$fieldnames,field_datatype = mydata1$data.type),

                            min))
distr.min <- data.frame(aggregate(x = mydata1[c("field_len",
                                  "fieldnames_len")],
                            by = list(fieldnames = mydata1$fieldnames,field_datatype = mydata1$data.type),

                            min))
```

The Result:

Data: distr.min

	fieldnames	field_datatype	field_len	fieldnames_len
1	firstname	character	0	9
2	gender	character	0	6
3	lastname	character	0	8
4	maritalstatus	character	0	13
5	middlename	character	0	10
6	birthdate	date	10	9
7	modifieddate	date	0	12
8	businessentityid	number	0	16
9	namestyle	number	0	9

Data: distr.mean

	fieldnames	field_datatype	field_len	fieldnames_len
1	firstname	character	5.9397696	9
2	gender	character	0.9992923	6
3	lastname	character	5.5420097	8
4	maritalstatus	character	0.9995611	13
5	middlename	character	1.0051214	10
6	birthdate	date	10.0000000	9
7	modifieddate	date	9.9702728	12
8	businessentityid	number	4.5746236	16
9	namestyle	number	0.9995627	9

Data: distr.max

	fieldnames	field_datatype	field_len	fieldnames_len
1	firstname	character	11	9
2	gender	character	1	6
3	lastname	character	17	8
4	maritalstatus	character	1	13
5	middlename	character	10	10
6	birthdate	date	10	9
7	modifieddate	date	10	12
8	businessentityid	number	5	16
9	namestyle	number	1	9

Then, these separate calculations are appended unto the unique list of elements. What is needed at the end is the data frame named *prelim_class_profile*

The Script:

```
### 12. Need to consolidate this information. So rename fields for each calcultion
###    with a meangingful name.ataset 2 & 3 from list. This is because we do not want repeated columns
names(distr.mean)[names(distr.mean)=="field_len"] <- "avg_field"
names(distr.mean)[names(distr.mean)=="fieldnames_len"] <- "avg_fieldnames"

names(distr.max)[names(distr.max)=="field_len"] <- "max_field"
names(distr.max)[names(distr.max)=="fieldnames_len"] <- "max_fieldnames"


names(distr.min)[names(distr.min)=="field_len"] <- "min_field"
names(distr.min)[names(distr.min)=="fieldnames_len"] <- "min_fieldnames"

### 13. Next order the fields according to the class name - so can join columns
distr.mean1<- distr.mean[order(distr.mean$fieldnames),]
distr.max1<- distr.max[order(distr.max$fieldnames),]
distr.min1<- distr.min[order(distr.min$fieldnames),]


### 14. Put all these calculated infot a list
distr.list <- list(distr.mean1, distr.max1,distr.min1)
View(distr.list)




### 15. Remove the class_name and data type for itsms 2 &3 in the list.
###     So we won't have redundant column names when we merge the data sets
for (i in seq_along(distr.list)[-1]) distr.list[[i]][,c('fieldnames')] <- NULL;
for (i in seq_along(distr.list)[-1]) distr.list[[i]][,c('field_datatype')] <- NULL;


### 16.So the pre-liminary dataset is almost done. Need to add the n-gram results
distr.list;
prelim_class_profile<-do.call(cbind,distr.list);


#### Create a list for future use
|
list.fieldnames <- lapply(seq_len(ncol(prelim_class_profile)), function(col) prelim_class_profile[,col])

View(prelim_class_profile)
str(prelim_class_profile)
View(list.fieldnames)
```

The Result:

	row.names	fieldnames	field_datatype	avg_field	avg_fieldnames	max_field	max_fieldnames	min_field	min_fieldnames
1	6	birthdate	date	10.0000000	9	10	9	10	9
2	8	businessentityid	number	4.5746236	16	5	16	0	16
3	1	firstname	character	5.9397696	9	11	9	0	9
4	2	gender	character	0.9992923	6	1	6	0	6
5	3	lastname	character	5.5420097	8	17	8	0	8
6	4	maritalstatus	character	0.9995611	13	1	13	0	13
7	5	middlename	character	1.0051214	10	10	10	0	10
8	7	modifieddate	date	9.9702728	12	10	12	0	12
9	9	namestyle	number	0.9995627	9	1	9	0	9

Before proceeding further, the dataset was divided into train (20% of records) and test (80% of records). Subsequently, from the train, 300 records were extracted for development purposes.

The Script:

```
### 17.Before can start on n-gram processing, first just take a sample of the data
### install.packages("tau")
library(tau)
### 18. Partition data to create elementTrain , elementTest1 and elementTest2
### install.packages("caret", dependencies = TRUE)
### remove.packages(c("ggplot2", "data.table"))
### install.packages('ggplot2', dependencies = TRUE)
### install.packages('data.table', dependencies = TRUE)

library(ggplot2)
library(caret)
library(psych)
library(plyr)
library(dplyr)

#split into training and test sets. We can also try with stratified sampling.

set.seed(185)
dfCleanedDat[,"test"] <- ifelse(runif(nrow(dfCleanedDat)) < 0.8, 1, 0)

#separate training and test sets

elementTrain <- dfCleanedDat[dfCleanedDat$train==0,]
elementTest <- dfCleanedDat[dfCleanedDat$train==1,]

#get column index of train flag. We are actually fetching the col num of this vriable here in the data.
trainColNum <- grep("train", names(elementTrain))

#remove train flag column from train and test sets
elementTrain<- elementTrain[,-trainColNum]
elementTest<- elementTest[,-trainColNum]

summary(elementTrain)
summary(elementTest)

### 19. Break train n test data into class info sets
View(elementTrain)

train.birthdate <- elementTrain[which(elementTrain$mapped_flg=='DIM_CUSTOMER_BirthDate' ), ]
train.gender <- elementTrain[which(elementTrain$mapped_flg=='DIM_CUSTOMER_Gender' ), ]
train.firstname <- elementTrain[which(elementTrain$mapped_flg=='DIM_CUSTOMER_FirstName' ), ]
train.lastname <- elementTrain[which(elementTrain$mapped_flg=='DIM_CUSTOMER_LastName' ), ]
train.namestyle <- elementTrain[which(elementTrain$mapped_flg=='DIM_CUSTOMER_NameStyle' ), ]
train.maritalstatus <- elementTrain[which(elementTrain$mapped_flg=='DIM_CUSTOMER_MaritalStatus' ), ]
train.middlename <- elementTrain[which(elementTrain$mapped_flg=='DIM_CUSTOMER_MiddleNAme' ), ]
train.unk <- elementTrain[which(elementTrain$mapped_flg=='UNK' ), ]

test.birthdate <- elementTest[which(elementTest$mapped_flg=='DIM_CUSTOMER_BirthDate' ), ]
test.gender <- elementTest[which(elementTest$mapped_flg=='DIM_CUSTOMER_Gender' ), ]
test.firstname <- elementTest[which(elementTest$mapped_flg=='DIM_CUSTOMER_FirstName' ), ]
test.lastname <- elementTest[which(elementTest$mapped_flg=='DIM_CUSTOMER_LastName' ), ]
test.namestyle <- elementTest[which(elementTest$mapped_flg=='DIM_CUSTOMER_NameStyle' ), ]
test.maritalstatus <- elementTest[which(elementTest$mapped_flg=='DIM_CUSTOMER_MaritalStatus' ), ]
test.middlename <- elementTest[which(elementTest$mapped_flg=='DIM_CUSTOMER_MiddleNAme' ), ]
test.unk <- elementTest[which(elementTest$mapped_flg=='UNK' ), ]

dev.birthdate<-train.birthdate[sample(nrow(train.birthdate), 300), ]
dev.gender <-train.gender [sample(nrow(train.gender ), 300), ]
dev.firstname<-train.firstname [sample(nrow(train.firstname ), 300), ]
dev.lastname<-train.lastname[sample(nrow(train.lastname), 300), ]
dev.middlename<-train.middlename[sample(nrow(train.middlename), 300), ]
dev.maritalstatus<-train.maritalstatus[sample(nrow(train.maritalstatus), 300), ]

dev.bday.profile<-prelim_class_profile[which(prelim_class_profile$fieldnames=='birthdate' ),]
dev.firstname.profile<-prelim_class_profile[which(prelim_class_profile$fieldnames=='firstname' ),]
dev.gender.profile<-prelim_class_profile[which(prelim_class_profile$fieldnames=='gender'),]
dev.lastname.profile<-prelim_class_profile[which(prelim_class_profile$fieldnames=='lastname' ),]
dev.middlename.profile<-prelim_class_profile[which(prelim_class_profile$fieldnames=='middlename' ),]
dev.maritalstatus.profile<-prelim_class_profile[which(prelim_class_profile$fieldnames=='maritalstatus' ),]
```

The Result:

```
> set.seed(185)
> dfCleanedDat[,"test"] <- ifelse(runif(nrow(dfCleanedDat)) < 0.8, 1, 0)
>
> #separate training and test sets
>
> elementTrain <- dfCleanedDat[dfCleanedDat$train==0,]
> elementTest <- dfCleanedDat[dfCleanedDat$train==1,]
>
> #get column index of train flag. We are actually fetching the col num of thi$
> trainColNum <- grep("train", names(elementTrain))
>
> #remove train flag column from train and test sets
> elementTrain<- elementTrain[,-trainColNum]
> elementTest<- elementTest[,-trainColNum]
>
> summary(elementTrain)
            fieldnames        fields          data.type
 businessentityid:3739   M      : 4024   character:16740
 lastname        :3695   O      : 3692   date     : 7248
 namestyle       :3694   F      : 1866   number   : 7433
 firstname       :3667   S      : 1738
 birthdate       :3658   A      :  259
 gender          :3644   L      :  245
 (Other)         :9324   (Other):19597
                  mapped_flg        test
 UNK                   :7329   Min.   :0.0000
 DIM_CUSTOMER_LastName :3695   1st Qu.:1.0000
 DIM_CUSTOMER_NameStyle:3694   Median :1.0000
 DIM_CUSTOMER_FirstName:3667   Mean   :0.7991
 DIM_CUSTOMER_BirthDate:3658   3rd Qu.:1.0000
 DIM_CUSTOMER_Gender   :3644   Max.   :1.0000
 (Other)               :5734
> summary(elementTest)
         fieldnames        fields          data.type
 modifieddate :14777   M      :16229   character:67054
 gender       :14725   O      :14598   date     :29324
 firstname    :14646   F      : 7418   number   :29059
 lastname     :14646   S      : 7099
 maritalstatus:14622   A      : 1022
 namestyle    :14600   L      :  996
 (Other)      :37421   (Other):78075
                      mapped_flg         test
 UNK                       :29236   Min.   :0.0000
 DIM_CUSTOMER_Gender       :14725   1st Qu.:1.0000
 DIM_CUSTOMER_FirstName    :14646   Median :1.0000
 DIM_CUSTOMER_LastName     :14646   Mean   :0.8021
 DIM_CUSTOMER_MaritalStatus:14622   3rd Qu.:1.0000
 DIM_CUSTOMER_NameStyle    :14600   Max.   :1.0000
 (Other)                   :22962
```

Next, the string content from all fields are concatenated and processed through the n-gram algorithm. This sample, shows the processing for three main fields: birthdate, gender and firstname.

The Script:

```
#20.combine all words in the data set per columns
### remove.packages(c("ngram"))
### install.packages("quanteda", dependencies = TRUE)
### install.packages("ngram", dependencies = TRUE)
### install.packages("reshape2", dependencies = TRUE)
library("ngram")
library("stringr")
library("quanteda")
library("reshape2")
library(stringdist)

na.zero <- function (x) {
    x[is.na(x)] <- 0
    return(x)
}

current.work<-dev.birthdate
field.len<- nchar(as.character(paste(current.work[,2], collapse=" ")))
workfile01<- paste(current.work[,2], collapse=" ")
workfile02<- preprocess(workfile01, case="upper")
workfile03<- str_replace_all(workfile02, "\\n", " ")
workfile04<- str_replace_all(workfile03, "\\\\", " ")
workfile05<- str_replace_all(workfile04, "\"", " ")
workfile06<- str_replace_all(workfile05, "/", " ")
workfile07<- str_replace_all(workfile06, "-", " ")
workfile08<- str_replace_all(workfile07, " ", "")
workfile09<- data.frame(do.call(rbind, strsplit(workfile08, "")))
workfile10<-t(workfile09)
n2bday<-ngrams(workfile10, n = 2)
n3bday<-ngrams(workfile10, n = 3)
n4bday<-ngrams(workfile10, n = 4)
n2workfile8<-ngrams(workfile08, n = 2)
n2workfile9<-ngrams(as.character(workfile09), n = 2)
bday_profile<-merge(n2bday, dev.bday.profile,all=TRUE)
View(n2bday)

current.work<-dev.gender
field.len<- nchar(as.character(paste(current.work[,2], collapse=" ")))
workfile01<- paste(current.work[,2], collapse=" ")
workfile02<- preprocess(workfile01, case="upper")
workfile03<- str_replace_all(workfile02, "\\n", " ")
workfile04<- str_replace_all(workfile03, "\\\\", " ")
workfile05<- str_replace_all(workfile04, "\"", " ")
workfile06<- str_replace_all(workfile05, "/", " ")
workfile07<- str_replace_all(workfile06, "-", " ")
workfile08<- str_replace_all(workfile07, " ", "")
workfile09<- data.frame(do.call(rbind, strsplit(workfile08, "")))
workfile10<-t(workfile09)
n2bday<-ngrams(workfile10, n = 2)
n3bday<-ngrams(workfile10, n = 3)
n4bday<-ngrams(workfile10, n = 4)
n2workfile8<-ngrams(workfile08, n = 2)
n2workfile9<-ngrams(as.character(workfile09), n = 2)
gender_profile<-merge(n2bday, dev.gender.profile,all=TRUE)


current.work<-dev.firstname
field.len<- nchar(as.character(paste(current.work[,2], collapse=" ")))
workfile01<- paste(current.work[,2], collapse=" ")
workfile02<- preprocess(workfile01, case="upper")
workfile03<- str_replace_all(workfile02, "\\n", " ")
workfile04<- str_replace_all(workfile03, "\\\\", " ")
workfile05<- str_replace_all(workfile04, "\"", " ")
workfile06<- str_replace_all(workfile05, "/", " ")
workfile07<- str_replace_all(workfile06, "-", " ")
workfile08<- str_replace_all(workfile07, " ", "")
workfile09<- data.frame(do.call(rbind, strsplit(workfile08, "")))
workfile10<-t(workfile09)
n2bday<-ngrams(workfile10, n = 2)
n3bday<-ngrams(workfile10, n = 3)
n4bday<-ngrams(workfile10, n = 4)
n2workfile8<-ngrams(workfile10, n = 2)
n2workfile9<-ngrams(as.character(workfile10), n = 2)
firstname_profile<-merge(n2bday, dev.firstname.profile,all=TRUE)
```

The Result:

a. Results of the n-gram breakdown for four, three and two character sequence.

	x
1	0_4_2_6
2	4_2_6_1
3	2_6_1_9
4	6_1_9_5
5	1_9_5_7
6	9_5_7_1
7	5_7_1_1
8	7_1_1_0
9	1_1_0_6
10	1_0_6_1
11	0_6_1_9
12	6_1_9_5
13	1_9_5_6
14	9_5_6_0
15	5_6_0_6
16	6_0_6_0
17	0_6_0_2
18	6_0_2_1
19	0_2_1_9
20	2_1_9_8
21	1_9_8_0
22	9_8_0_1
23	8_0_1_1
24	0_1_1_0

	x
1	0_4_2
2	4_2_6
3	2_6_1
4	6_1_9
5	1_9_5
6	9_5_7
7	5_7_1
8	7_1_1
9	1_1_0
10	1_0_6
11	0_6_1
12	6_1_9
13	1_9_5
14	9_5_6
15	5_6_0
16	6_0_6
17	0_6_0
18	6_0_2
19	0_2_1
20	2_1_9
21	1_9_8
22	9_8_0
23	8_0_1
24	0_1_1

	x
1	0_4
2	4_2
3	2_6
4	6_1
5	1_9
6	9_5
7	5_7
8	7_1
9	1_1
10	1_0
11	0_6
12	6_1
13	1_9
14	9_5
15	5_6
16	6_0
17	0_6
18	6_0
19	0_2
20	2_1
21	1_9
22	9_8
23	8_0
24	0_1

b. Final n-gram combined with profile information (two character n-gram only)

Data: bday_profile

	x	fieldnames	field_datatype	avg_field	avg_fieldnames	max_field	max_fieldnames	min_field	min_fieldnames
1	0_4	birthdate	date	10	9	10	9	10	9
2	4_2	birthdate	date	10	9	10	9	10	9
3	2_6	birthdate	date	10	9	10	9	10	9
4	6_1	birthdate	date	10	9	10	9	10	9
5	1_9	birthdate	date	10	9	10	9	10	9
6	9_5	birthdate	date	10	9	10	9	10	9
7	5_7	birthdate	date	10	9	10	9	10	9

Data: gender_profile

	x	fieldnames	field_datatype	avg_field	avg_fieldnames	max_field	max_fieldnames	min_field	min_fieldnames
1	M_M	gender	character	0.9992923	6	1	6	0	6
2	M_M	gender	character	0.9992923	6	1	6	0	6
3	M_M	gender	character	0.9992923	6	1	6	0	6
4	M_F	gender	character	0.9992923	6	1	6	0	6
5	F_M	gender	character	0.9992923	6	1	6	0	6
6	M_F	gender	character	0.9992923	6	1	6	0	6
7	F_M	gender	character	0.9992923	6	1	6	0	6

Data: firstname_profile

	x	fieldnames	field_datatype	avg_field	avg_fieldnames	max_field	max_fieldnames	min_field	min_fieldnames
1	S_A	firstname	character	5.93977	9	11	9	0	9
2	A_R	firstname	character	5.93977	9	11	9	0	9
3	R_A	firstname	character	5.93977	9	11	9	0	9
4	A_E	firstname	character	5.93977	9	11	9	0	9
5	E_M	firstname	character	5.93977	9	11	9	0	9
6	M_M	firstname	character	5.93977	9	11	9	0	9
7	M_A	firstname	character	5.93977	9	11	9	0	9

Finally, the three datasets are combined to provide a combination of n-gram distributions against all three elements:

The Script:

```
ds1<-rbind(as.matrix(bday_profile),as.matrix(firstname_profile),as.matrix(gender_profile))
#,as.matrix(lastname_profile))
ds2<-as.data.frame(ds1)
```

The Result:

Data: ds2

	x	fieldnames	field_datatype	avg_field	avg_fieldnames	max_field	max_fieldnames	min_field	min_fieldnames
2395	0_1	birthdate	date	10	9	10	9	10	9
2396	1_1	birthdate	date	10	9	10	9	10	9
2397	1_9	birthdate	date	10	9	10	9	10	9
2398	9_5	birthdate	date	10	9	10	9	10	9
2399	5_0	birthdate	date	10	9	10	9	10	9
2400	S_A	firstname	character	5.93977	9	11	9	0	9
2401	A_R	firstname	character	5.93977	9	11	9	0	9
2402	R_A	firstname	character	5.93977	9	11	9	0	9
2403	A_E	firstname	character	5.93977	9	11	9	0	9
2404	E_M	firstname	character	5.93977	9	11	9	0	9

DESCRIPTIVE ANALYSIS

Now we can start analysing the data set. The hypothesis is that with the distribution of characters used by the elements coupled with the element and element content information the models should be able to predict the mapped field.

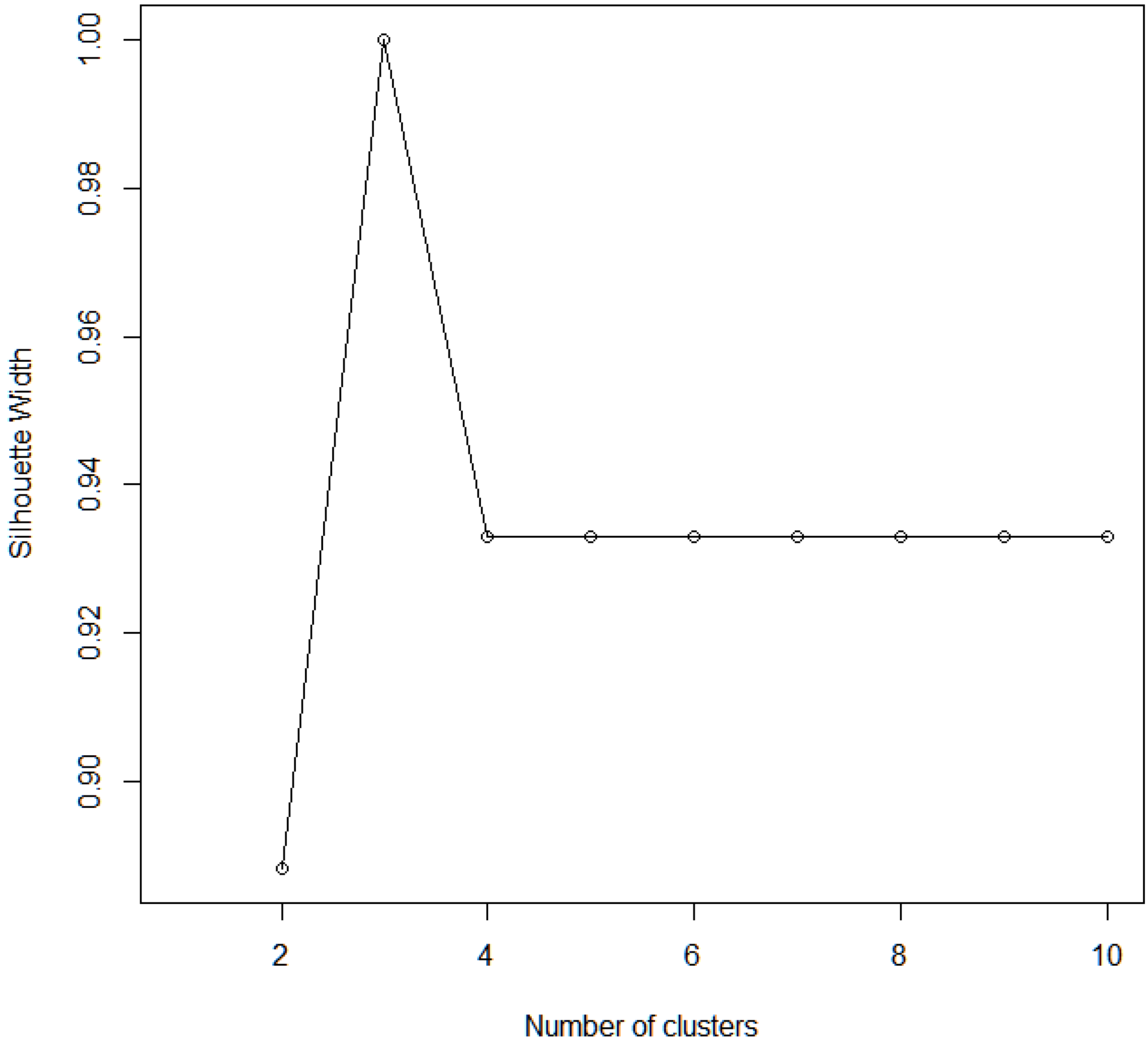
To support the hypothesis, we look at the performance of a cluster analysis using the data set.

A *scree plot* is run to determine if the number of cluster can be identified. Subsequently, as *ggplot* will visualize the proximity of the clusters identified.

The Script:

```
#21.Install Clustering Packages

library(ngram)
library(stringr)
library(tau)
library(tm)
library(stringdist)


### install.packages("Rtsne", dependencies = TRUE)
### install.packages("ISLR", dependencies = TRUE)
### install.packages("cluster", dependencies = TRUE)
### install.packages("labeling", dependencies = TRUE)
### install.packages("clara", dependencies = TRUE)

library(dplyr)    # for data cleaning
library(ISLR)
library(cluster) # for gower similarity and pam
library(Rtsne)    # for t-SNE plot
library(ggplot2) # for visualization

#22.Run Distances Checks
gower_dist <- daisy(ds2[, -1],
                    metric = "gower",
                    type = list(logratio = 3))

summary(gower_dist)
tsne_obj <- Rtsne(gower_dist, is_distance = TRUE)
sil_width <- c(NA)

for(i in 2:10){
  pam_fit <- pam(gower_dist,
                 diss = TRUE,
                 k = i)
  sil_width[i] <- pam_fit$silinfo$avg.width}


# Plot sihouette width (higher is better)
plot(1:10, sil_width,
     xlab = "Number of clusters",
     ylab = "Silhouette Width")
lines(1:10, sil_width)
```

The Result:

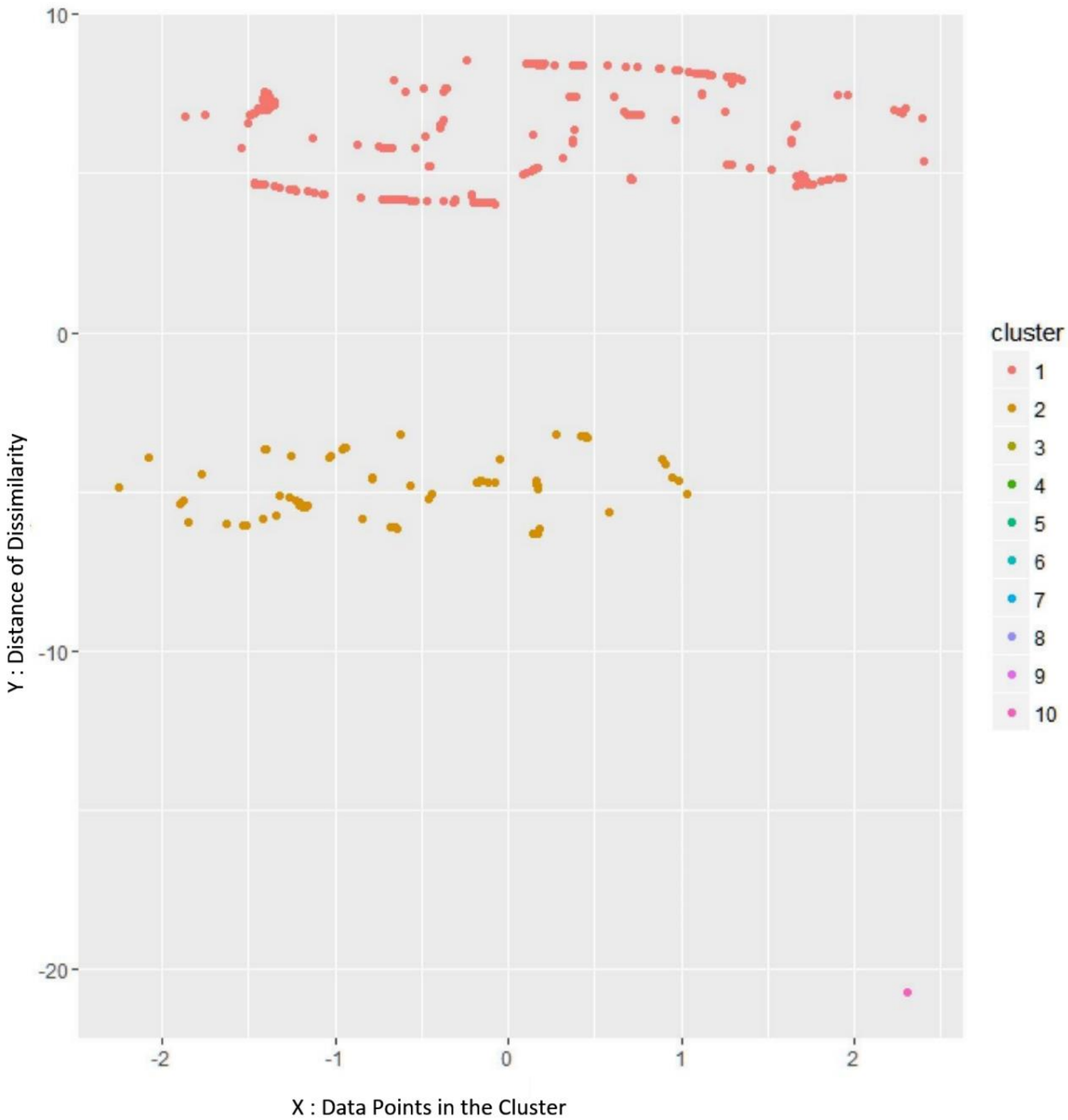

Silhouette Width
1.00
0.98
0.96
0.94
0.92
0.90
2
4
6
8
10
Number of clusters

Next, a visualization of the distribution of the information.

The Script:

```
tsne_data <- tsne_obj$Y %>%
  data.frame() %>%
  setNames(c("X", "Y")) %>%
  mutate(cluster = factor(pam_fit$clustering),
         name = ds2$x)

ggplot(aes(x = X, y = Y), data = tsne_data) +  geom_point(aes(color = cluster))
```

The Result:



The cluster visualization defines 10 which can be identified in the below data set.

	X	Y	cluster	name
4452	4.509744092	20.942314	3	M_F
4453	4.509744093	20.942314	3	F_F
4454	4.509744094	20.942314	3	F_M
4455	4.509744092	20.942314	3	M_M
4456	4.509744092	20.942314	3	M_F
4457	4.509744092	20.942314	3	F_M
4458	4.509744090	20.942314	3	M_F
4459	4.509744091	20.942314	3	F_F
4460	4.509744092	20.942314	3	F_M
4461	4.509744091	20.942314	3	M_M
4462	4.509744092	20.942314	3	M_F
4463	4.509744088	20.942314	3	F_F
4464	4.509744089	20.942314	4	F_F
4465	4.509744092	20.942314	5	F_F
4466	4.509744092	20.942314	6	F_M
4467	4.509744092	20.942314	7	M_M
4468	4.509744088	20.942314	8	M_M
4469	4.509744092	20.942314	9	M_F
4470	4.509599393	20.941983	10	F_F

PREDICTION

The prediction is generated using the kNN algorithm made available by the RWeka package and uses the IBk function.

The Script:

```
#*************************************************
#    PREDICTION BEGININGS
#*************************************************

devPredprelim<-merge(ds2  , mydata1Pred_grp,all=TRUE)
View(devPredprelim)
table(devPredprelim$mapped_flg) # look at the frequencies for the left variable
table(devPredprelim$mapped_flg)/nrow(devPredprelim) # look at percentages for the left variable
devPredprelim[,"train"] <- ifelse(runif(nrow(devPredprelim)) < 0.8, 1, 0)

#separate training and test sets
elementDevTrain <- devPredprelim[devPredprelim$train==1,]
elementDevTest <- devPredprelim[devPredprelim$train==0,]


#get column index of train flag. We are actually fetching the col num of this vriable here in the dat
trainColNum <- grep("train", names(elementDevTrain ))

#remove train flag column from train and test sets
elementDevTrain <- elementDevTrain[,-trainColNum]
elementDevTest<- elementDevTest[,-trainColNum]
elementDevTrain2<-elementDevTrain[complete.cases(elementDevTrain),]


#*************************************************
#24.Prediction using KNN
#*************************************************
library(class)
library(stats)
library(gmodels)

elementDevTrain2.labels <- elementDevTrain2[,10]
elementDevTest[is.na(elementDevTest)] <- as.numeric(0)
elementDevTrain2[is.na(elementDevTrain2)] <- as.numeric(0)
elementDevTrain2.class<-as.factor(elementDevTrain2[,10])

elementDevTest<-as.data.frame(elementDevTest)
elementDevTrain2<-as.data.frame(elementDevTrain2)
elementDevTrain2.class<-as.data.frame(elementDevTrain2.class)

library("RWeka")

classifier <- IBk(as.factor(mapped_flg) ~., data = elementDevTrain2,control = Weka_control(K = 9))
```

The Result:

```
> table(devPredprelim$mapped_flg) # look at the frequencies for the left variable

    DIM_CUSTOMER_BirthDate      DIM_CUSTOMER_FirstName
                      2399                        1772
       DIM_CUSTOMER_Gender        DIM_CUSTOMER_LastName
                       299                           1
DIM_CUSTOMER_MaritalStatus     DIM_CUSTOMER_MiddleNAme
                         1                           1
    DIM_CUSTOMER_NameStyle                         UNK
                         1                           2
> table(devPredprelim$mapped_flg)/nrow(devPredprelim) # look at percentages for the left variable

    DIM_CUSTOMER_BirthDate      DIM_CUSTOMER_FirstName
              0.5359696157                0.3958891868
       DIM_CUSTOMER_Gender        DIM_CUSTOMER_LastName
              0.0668007149                0.0002234138
DIM_CUSTOMER_MaritalStatus     DIM_CUSTOMER_MiddleNAme
              0.0002234138                0.0002234138
    DIM_CUSTOMER_NameStyle                         UNK
              0.0002234138                0.0004468275
```

Data: devPredprelim

	fieldnames	x	field_datatype	avg_field	avg_fieldnames	max_field	max_fieldnames	min_field	min_fieldnames	mapped_flg	train
4457	gender	F_M	character	0.9992923	6	1	6	0	6	DIM_CUSTOMER_Gender	1
4458	gender	M_F	character	0.9992923	6	1	6	0	6	DIM_CUSTOMER_Gender	0
4459	gender	F_F	character	0.9992923	6	1	6	0	6	DIM_CUSTOMER_Gender	1
4460	gender	F_M	character	0.9992923	6	1	6	0	6	DIM_CUSTOMER_Gender	0
4461	gender	M_M	character	0.9992923	6	1	6	0	6	DIM_CUSTOMER_Gender	1
4462	gender	M_F	character	0.9992923	6	1	6	0	6	DIM_CUSTOMER_Gender	1
4463	gender	F_F	character	0.9992923	6	1	6	0	6	DIM_CUSTOMER_Gender	1
4464	gender	F_F	character	0.9992923	6	1	6	0	6	DIM_CUSTOMER_Gender	1
4465	gender	F_F	character	0.9992923	6	1	6	0	6	DIM_CUSTOMER_Gender	1
4466	gender	F_M	character	0.9992923	6	1	6	0	6	DIM_CUSTOMER_Gender	1
4467	gender	M_M	character	0.9992923	6	1	6	0	6	DIM_CUSTOMER_Gender	0
4468	gender	M_M	character	0.9992923	6	1	6	0	6	DIM_CUSTOMER_Gender	1
4469	gender	M_F	character	0.9992923	6	1	6	0	6	DIM_CUSTOMER_Gender	0
4470	gender	F_F	character	0.9992923	6	1	6	0	6	DIM_CUSTOMER_Gender	1
4471	businessentityid	NA	NA	NA	NA	NA	NA	NA	NA	UNK	0
4472	lastname	NA	NA	NA	NA	NA	NA	NA	NA	DIM_CUSTOMER_LastName	1
4473	maritalstatus	NA	NA	NA	NA	NA	NA	NA	NA	DIM_CUSTOMER_MaritalStatus	1
4474	middlename	NA	NA	NA	NA	NA	NA	NA	NA	DIM_CUSTOMER_MiddleNAme	1
4475	modifieddate	NA	NA	NA	NA	NA	NA	NA	NA	UNK	1
4476	namestyle	NA	NA	NA	NA	NA	NA	NA	NA	DIM_CUSTOMER_NameStyle	1

VALIDATION

Finally, the performance of the IBk function is validated by checking the confusion matrix using the same RWeka package.

The Script:

```
evaluate_Weka_classifier(classifier,newdata = elementDevTest)
```

The Result

```
=== Summary ===

Correctly Classified Instances         921              100      %
Incorrectly Classified Instances         0                0      %
Kappa statistic                          1
Mean absolute error                      0
Root mean squared error                  0
Relative absolute error                  0.007 %
Root relative squared error              0.01  %
Coverage of cases (0.95 level)         100     %
Mean rel. region size (0.95 level)      12.5   %
Total Number of Instances              921

=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   <-- classified as
 492   0   0   0   0   0   0   0 |   a = DIM_CUSTOMER_BirthDate
   0 367   0   0   0   0   0   0 |   b = DIM_CUSTOMER_FirstName
   0   0  62   0   0   0   0   0 |   c = DIM_CUSTOMER_Gender
   0   0   0   0   0   0   0   0 |   d = DIM_CUSTOMER_LastName
   0   0   0   0   0   0   0   0 |   e = DIM_CUSTOMER_MaritalStatus
   0   0   0   0   0   0   0   0 |   f = DIM_CUSTOMER_MiddleNAme
   0   0   0   0   0   0   0   0 |   g = DIM_CUSTOMER_NameStyle
   0   0   0   0   0   0   0   0 |   h = UNK
```

The results above is overfitting to the data set and therefore is 100 % accurate.

This documentation serves only as a step-by-step guide to analyse the capability of machine learning in the identification of a matching target field. Continued analysis needs to be carried out to further uncover the strengths and weaknesses of this approach.