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** ........................................................................................................... 3
- **SYSTEM CONFIGURATION** .......................................................................................... 3
- **APPLICATION REQUIREMENT** ..................................................................................... 3
- **WORKFLOW DIAGRAM** ................................................................................................. 4
- **DATA PREPARATION** .................................................................................................... 4
- **DATA CLEANSING** ....................................................................................................... 7
  - Importing and Parsing XML through R ........................................................................... 7
  - Cleansing of Parsed Elements ....................................................................................... 8
- **DESCRIPTIVE ANALYSIS** ............................................................................................... 21
- **PREDICTION** ................................................................................................................ 26
- **VALIDATION** ................................................................................................................ 28
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.

![System Configuration](image)

APPLICATION REQUIREMENT

Application used for this analysis are:

![Application Requirement](image)
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
drop table TMP_DISS_TBL;

SELECT   p.[BusinessEntityID]
  ,p.[FirstName]
  ,p.[MiddleName]
  ,p.[LastName]
  ,CONVERT(datetime, REPLACE([IndividualSurvey].[ref].[value])(N'declare default element namespace "
  BirthDate(l)', 'nvarchar(20)') ,',', ''), 101) AS [BirthDate]
  ,,[IndividualSurvey].[ref].[value](N'declare default element namespace "
  MaritalStatus(l)', 'nvarchar(1)') AS [MaritalStatus]
  ,,[IndividualSurvey].[ref].[value](N'declare default element namespace "
  Gender(l)', '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/Demographics";
   /ci:Demographics' } 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');
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 step requires the RVEST and XML packages to be installed.

The Script:

```r
data <- read_xml("C:\dissertation\scripts\02b_ext_in_xml.xml")
people <- data$persons
fieldnames <- data$fieldnames
fields <- data$fields
```

The Result:

<table>
<thead>
<tr>
<th></th>
<th>businessentityid</th>
<th>firstname</th>
<th>middlename</th>
<th>lastname</th>
<th>namestyle</th>
<th>birthdate</th>
<th>maritalstatus</th>
<th>gender</th>
<th>modifieddate</th>
<th>businessentityid</th>
<th>firstname</th>
<th>middlename</th>
<th>lastname</th>
<th>namestyle</th>
<th>birthdate</th>
</tr>
</thead>
</table>
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.

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

![CSV Data Import into Excel](image)

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="totalpurchasetotal","number",
IF(B2="totalchildren","number",
IF(B2="rowguid","raw",
IF(B2="personotype","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="individualsex","number",
IF(B2="homeownerflag", "number",
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="marriedstatus","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 compute faster in R.

**Formula Column E:**

```
=IF(B2="namestyle","DIM_CUSTOMER/NameStyle",
IF(B2="occupation","DIM_CUSTOMER/EnglishOccupation",
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",
```

```
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.

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:

```r
### 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:

```r
> ### 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)
> [1] 156858   4
> summary(dfCleanedDat)

<table>
<thead>
<tr>
<th>fieldnames</th>
<th>fields</th>
<th>data.type</th>
</tr>
</thead>
<tbody>
<tr>
<td>gender</td>
<td>18369</td>
<td>M</td>
</tr>
<tr>
<td>modifiedDate</td>
<td>18367</td>
<td>0</td>
</tr>
<tr>
<td>lastname</td>
<td>18341</td>
<td>F</td>
</tr>
<tr>
<td>firstname</td>
<td>18313</td>
<td>S</td>
</tr>
<tr>
<td>namestyle</td>
<td>18294</td>
<td>A</td>
</tr>
<tr>
<td>maritalstatus</td>
<td>18227</td>
<td>L</td>
</tr>
<tr>
<td>(Other)</td>
<td>46947</td>
<td>(Other)</td>
</tr>
</tbody>
</table>

mapped_flg

<table>
<thead>
<tr>
<th>fieldnames</th>
<th>fields</th>
<th>data.type</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNK</td>
<td>36565</td>
<td></td>
</tr>
<tr>
<td>DIM_CUSTOMER_Gender</td>
<td>18369</td>
<td></td>
</tr>
<tr>
<td>DIM_CUSTOMER_LastName</td>
<td>18341</td>
<td></td>
</tr>
<tr>
<td>DIM_CUSTOMER_FirstName</td>
<td>18313</td>
<td></td>
</tr>
<tr>
<td>DIM_CUSTOMER_NameStyle</td>
<td>18294</td>
<td></td>
</tr>
<tr>
<td>DIM_CUSTOMER_MaritalStatus</td>
<td>18227</td>
<td></td>
</tr>
<tr>
<td>(Other)</td>
<td>28749</td>
<td></td>
</tr>
</tbody>
</table>

'data.frame': 156858 obs. of 4 variables:
$ fieldnames: Factor w/ 9 levels "birthDate","businessentityid",...: 2 3 7 5 9 11
$ fields : Factor w/ 27609 levels ",","0","01-Apr",...: 15277 27071 27266 27334
$ 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"
[2] "DIM_CUSTOMER_Gender" "DIM_CUSTOMER_LastName"
[3] "DIM_CUSTOMER_MaritalStatus" "DIM_CUSTOMER_MiddleName"
[4] "DIM_CUSTOMER_NameStyle" "UNK"
```
Next, the profile information of elements and element contents need to be concatenated to the data frame.

<table>
<thead>
<tr>
<th>fieldnames</th>
<th>fields</th>
<th>data.type</th>
<th>mapped_flg</th>
</tr>
</thead>
<tbody>
<tr>
<td>businessentityid</td>
<td>1708</td>
<td>number</td>
<td>UNK</td>
</tr>
<tr>
<td>firstname</td>
<td>Linda</td>
<td>character</td>
<td>DIM_CUSTOMER_FirstName</td>
</tr>
<tr>
<td>middlename</td>
<td>R.</td>
<td>character</td>
<td>DIM_CUSTOMER_MiddleName</td>
</tr>
<tr>
<td>lastname</td>
<td>Rousey</td>
<td>character</td>
<td>DIM_CUSTOMER_LastName</td>
</tr>
<tr>
<td>namestyle</td>
<td>0</td>
<td>number</td>
<td>DIM_CUSTOMER_NameStyle</td>
</tr>
<tr>
<td>birthdate</td>
<td>04/07/1947</td>
<td>date</td>
<td>DIM_CUSTOMER_BirthDate</td>
</tr>
<tr>
<td>maritalstatus</td>
<td>M</td>
<td>character</td>
<td>DIM_CUSTOMER_MaritalStatus</td>
</tr>
<tr>
<td>gender</td>
<td>F</td>
<td>character</td>
<td>DIM_CUSTOMER_Gender</td>
</tr>
<tr>
<td>modifieddate</td>
<td>04/07/1947</td>
<td>date</td>
<td>UNK</td>
</tr>
<tr>
<td>businessentityid</td>
<td>1715</td>
<td>number</td>
<td>UNK</td>
</tr>
<tr>
<td>firstname</td>
<td>Justine</td>
<td>character</td>
<td>DIM_CUSTOMER_FirstName</td>
</tr>
<tr>
<td>middlename</td>
<td>J.</td>
<td>character</td>
<td>DIM_CUSTOMER_MiddleName</td>
</tr>
<tr>
<td>lastname</td>
<td>Ryan</td>
<td>character</td>
<td>DIM_CUSTOMER_LastName</td>
</tr>
<tr>
<td>namestyle</td>
<td>0</td>
<td>number</td>
<td>DIM_CUSTOMER_NameStyle</td>
</tr>
<tr>
<td>birthdate</td>
<td>11/11/1947</td>
<td>date</td>
<td>DIM_CUSTOMER_BirthDate</td>
</tr>
<tr>
<td>maritalstatus</td>
<td>S</td>
<td>character</td>
<td>DIM_CUSTOMER_MaritalStatus</td>
</tr>
<tr>
<td>gender</td>
<td>F</td>
<td>character</td>
<td>DIM_CUSTOMER_Gender</td>
</tr>
<tr>
<td>modifieddate</td>
<td>11/11/1947</td>
<td>date</td>
<td>UNK</td>
</tr>
<tr>
<td>businessentityid</td>
<td>1722</td>
<td>number</td>
<td>UNK</td>
</tr>
<tr>
<td>firstname</td>
<td>Mandar</td>
<td>character</td>
<td>DIM_CUSTOMER_FirstName</td>
</tr>
<tr>
<td>lastname</td>
<td>Samant</td>
<td>character</td>
<td>DIM_CUSTOMER_LastName</td>
</tr>
<tr>
<td>namestyle</td>
<td>0</td>
<td>number</td>
<td>DIM_CUSTOMER_NameStyle</td>
</tr>
</tbody>
</table>
The Script:

```r
### 8. Make a a working data frame copy i.e. : mydata1 of unique information
mydata1 <- dfCleanedDat
require(dplyr)
mydata1Fred_grp <- unique(mydata1[,c("fieldnames", "mapped_flg")])

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

<table>
<thead>
<tr>
<th>fieldnames</th>
<th>field_datatype</th>
<th>field_len</th>
<th>fieldnames_len</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 firstname</td>
<td>character</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>2 gender</td>
<td>character</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>3 lastname</td>
<td>character</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>4 maritalstatus</td>
<td>character</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>5 middlename</td>
<td>character</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>6 birthdate</td>
<td>date</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>7 modifieddate</td>
<td>date</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>8 businessentityid</td>
<td>number</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>9 namestyle</td>
<td>number</td>
<td>0</td>
<td>9</td>
</tr>
</tbody>
</table>

#### Data: distr.mean

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

#### Data: distr.max

<table>
<thead>
<tr>
<th>fieldnames</th>
<th>field_datatype</th>
<th>field_len</th>
<th>fieldnames_len</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 firstname</td>
<td>character</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>2 gender</td>
<td>character</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>3 lastname</td>
<td>character</td>
<td>17</td>
<td>8</td>
</tr>
<tr>
<td>4 maritalstatus</td>
<td>character</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>5 middlename</td>
<td>character</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>6 birthdate</td>
<td>date</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>7 modifieddate</td>
<td>date</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>8 businessentityid</td>
<td>number</td>
<td>5</td>
<td>16</td>
</tr>
<tr>
<td>9 namestyle</td>
<td>number</td>
<td>1</td>
<td>9</td>
</tr>
</tbody>
</table>
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:

```r
### 12. Need to consolidate this information. So rename fields for each calculation
### with a meaningful name at set 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.max1 <- distr.max[order(distr.max$fieldnames),]
distr.min1 <- distr.min[order(distr.min$fieldnames),]

distr.list <- list(distr.mean1, distr.max1, distr.min1)
View(distr.list)

### 15. Remove the class_name and data type for items 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]][, "fieldnames"] <- NULL;
for (i in seq_along(distr.list)[[-1]]) distr.list[[i]][, "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:

<table>
<thead>
<tr>
<th>row_names</th>
<th>fieldnames</th>
<th>field datatypes</th>
<th>avg_field</th>
<th>avg fieldnames</th>
<th>max field</th>
<th>max fieldnames</th>
<th>min field</th>
<th>min fieldnames</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>birthdate</td>
<td>date</td>
<td>10.000000</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>businessentityid</td>
<td>number</td>
<td>4.5746236</td>
<td>16</td>
<td>5</td>
<td>16</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>firstname</td>
<td>character</td>
<td>5.9997694</td>
<td>9</td>
<td>11</td>
<td>9</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>gender</td>
<td>character</td>
<td>0.9999323</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>lastname</td>
<td>character</td>
<td>5.5420097</td>
<td>8</td>
<td>17</td>
<td>8</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>maritalstatus</td>
<td>character</td>
<td>0.9995611</td>
<td>13</td>
<td>1</td>
<td>13</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>7</td>
<td>middlename</td>
<td>character</td>
<td>1.0001214</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>modifieddate</td>
<td>date</td>
<td>9.9702728</td>
<td>12</td>
<td>10</td>
<td>12</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>9</td>
<td>namestyle</td>
<td>number</td>
<td>0.9995627</td>
<td>9</td>
<td>1</td>
<td>9</td>
<td>0</td>
<td>9</td>
</tr>
</tbody>
</table>

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:

```r
## 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)
## install.packages("ggplot2", "data.table")
## install.packages("ggplot2", dependencies = TRUE)
library(ggplot2)
library(caret)
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]

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

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

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

dev.birthday<-train.birthday[sample(nrow(train.birthday), 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$filenames=='birthday'),]
dev.firstname.profile<-prelim_class_profile[which(prelim_class_profile$filenames=='firstname'),]
dev.gender.profile<-prelim_class_profile[which(prelim_class_profile$filenames=='gender'),]
dev.lastname.profile<-prelim_class_profile[which(prelim_class_profile$filenames=='lastname'),]
dev.middlename.profile<-prelim_class_profile[which(prelim_class_profile$filenames=='middle'),]
dev.maritalstatus.profile<-prelim_class_profile[which(prelim_class_profile$filenames=='maritalstatus'),]
```
The Result:

```r
> 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$ 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
```

```r
> 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
```

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:

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

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

current.workday <- dev.birthday
field.len <- nchar(as.character(paste(current.workday[,2], collapse = " ")))
workfile10 <- paste(current.workday[,2], collapse = " ")
workfile08 <- str_replace_all(workfile02, \"\n\", " ")
workfile04 <- str_replace_all(workfile03, \"\n\", " ")
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)
n2workfile08 <- ngrams(workfile08, n = 2)
n2workfile09 <- as.character(workfile09)

bday_profile <- merge(n2bday, dev.birthday, all = TRUE)
View(n2bday)

current.workday <- dev.gender
field.len <- nchar(as.character(paste(current.workday[,2], collapse = " ")))
workfile10 <- paste(current.workday[,2], collapse = " ")
workfile08 <- str_replace_all(workfile02, \"\n\", " ")
workfile04 <- str_replace_all(workfile03, \"\n\", " ")
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)
n2workfile08 <- ngrams(workfile08, n = 2)
n2workfile09 <- as.character(workfile09)

gender_profile <- merge(n2bday, dev.gender, all = TRUE)


current.workday <- dev.firstname
field.len <- nchar(as.character(paste(current.workday[,2], collapse = " ")))
workfile10 <- paste(current.workday[,2], collapse = " ")
workfile08 <- str_replace_all(workfile02, \"\n\", " ")
workfile04 <- str_replace_all(workfile03, \"\n\", " ")
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)
n2workfile08 <- ngrams(workfile08, n = 2)
n2workfile09 <- as.character(workfile09)

firstname_profile <- merge(n2bday, dev.firstname, all = TRUE)
```
The Result:

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

b. Final n-gram combined with profile information (two character n-gram only)
Finally, the three datasets are combined to provide a combination of n-gram distributions against all three elements:

The Script:
```r
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:
```
<table>
<thead>
<tr>
<th>x</th>
<th>fieldnames</th>
<th>field_datatype</th>
<th>avg_field</th>
<th>avg_fieldnames</th>
<th>max_field</th>
<th>max_fieldnames</th>
<th>min_field</th>
<th>min_fieldnames</th>
</tr>
</thead>
<tbody>
<tr>
<td>2395</td>
<td>0.1</td>
<td>birthdate</td>
<td>date</td>
<td>10</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>2396</td>
<td>1.1</td>
<td>birthdate</td>
<td>date</td>
<td>10</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>2397</td>
<td>1.9</td>
<td>birthdate</td>
<td>date</td>
<td>10</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>2398</td>
<td>8.5</td>
<td>birthdate</td>
<td>date</td>
<td>10</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>2399</td>
<td>5.0</td>
<td>birthdate</td>
<td>date</td>
<td>10</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>2400</td>
<td>S_A</td>
<td>firstname</td>
<td>character</td>
<td>5.93977</td>
<td>9</td>
<td>11</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>2401</td>
<td>A_R</td>
<td>firstname</td>
<td>character</td>
<td>5.93977</td>
<td>9</td>
<td>11</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>2402</td>
<td>B_A</td>
<td>firstname</td>
<td>character</td>
<td>5.93977</td>
<td>9</td>
<td>11</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>2403</td>
<td>A_E</td>
<td>firstname</td>
<td>character</td>
<td>5.93977</td>
<td>9</td>
<td>11</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>2404</td>
<td>B_M</td>
<td>firstname</td>
<td>character</td>
<td>5.93977</td>
<td>9</td>
<td>11</td>
<td>9</td>
<td>0</td>
</tr>
</tbody>
</table>
```

**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$silinfo$sav.width
}

# Plot silhouette width (higher is better)
plot(1:10, sil_width,
     xlab = "Number of clusters",
     ylab = "Silhouette Width")
lines(1:10, sil_width)
The Result:
Next, a visualization of the distribution of the information.

The Script:

```r
# The visualization code

tsnne_data <- tsnne_obj$Y
  data.frame() 
  setNames(c("X", "Y")) 
  mutate(cluster = factor(pam_fit$clustering),
         name = ds2$x)

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

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

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>cluster</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>4452</td>
<td>4.509744092</td>
<td>20.942314</td>
<td>3</td>
</tr>
<tr>
<td>4453</td>
<td>4.509744093</td>
<td>20.942314</td>
<td>3</td>
</tr>
<tr>
<td>4454</td>
<td>4.509744094</td>
<td>20.942314</td>
<td>3</td>
</tr>
<tr>
<td>4455</td>
<td>4.509744092</td>
<td>20.942314</td>
<td>3</td>
</tr>
<tr>
<td>4456</td>
<td>4.509744092</td>
<td>20.942314</td>
<td>3</td>
</tr>
<tr>
<td>4457</td>
<td>4.509744092</td>
<td>20.942314</td>
<td>3</td>
</tr>
<tr>
<td>4458</td>
<td>4.509744090</td>
<td>20.942314</td>
<td>3</td>
</tr>
<tr>
<td>4459</td>
<td>4.509744091</td>
<td>20.942314</td>
<td>3</td>
</tr>
<tr>
<td>4460</td>
<td>4.509744092</td>
<td>20.942314</td>
<td>3</td>
</tr>
<tr>
<td>4461</td>
<td>4.509744091</td>
<td>20.942314</td>
<td>3</td>
</tr>
<tr>
<td>4462</td>
<td>4.509744092</td>
<td>20.942314</td>
<td>3</td>
</tr>
<tr>
<td>4463</td>
<td>4.509744088</td>
<td>20.942314</td>
<td>3</td>
</tr>
<tr>
<td>4464</td>
<td>4.509744089</td>
<td>20.942314</td>
<td>4</td>
</tr>
<tr>
<td>4465</td>
<td>4.509744092</td>
<td>20.942314</td>
<td>5</td>
</tr>
<tr>
<td>4466</td>
<td>4.509744092</td>
<td>20.942314</td>
<td>6</td>
</tr>
<tr>
<td>4467</td>
<td>4.509744092</td>
<td>20.942314</td>
<td>7</td>
</tr>
<tr>
<td>4468</td>
<td>4.509744088</td>
<td>20.942314</td>
<td>8</td>
</tr>
<tr>
<td>4469</td>
<td>4.509744092</td>
<td>20.942314</td>
<td>9</td>
</tr>
<tr>
<td>4470</td>
<td>4.509599393</td>
<td>20.941983</td>
<td>10</td>
</tr>
</tbody>
</table>
The prediction is generated using the kNN algorithm made available by the RWeka package and uses the IBk function.

The Script:

```r
# PREDICTION BEGINNINGS

deVPredprelim<-merge(ds2, mydata1Pred_grp, all=TRUE)
View(deVPredprelim)
table(deVPredprelim$mapped_flg) # look at frequencies for the left variable
table(deVPredprelim$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,]

deVPredprelim <- 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:

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

<table>
<thead>
<tr>
<th>Dim_Customer_BirthDate</th>
<th>Dim_Customer_FirstName</th>
</tr>
</thead>
<tbody>
<tr>
<td>2399</td>
<td>1772</td>
</tr>
<tr>
<td>Dim_Customer_Gender</td>
<td>Dim_Customer_LastName</td>
</tr>
<tr>
<td>129</td>
<td>1</td>
</tr>
<tr>
<td>Dim_Customer_NameStyle</td>
<td>Dim_Customer_MiddleName</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

> table(devPredprelim$mapped_flg)/nrow(devPredprelim) # look at percentages for the left variable

<table>
<thead>
<tr>
<th>Dim_Customer_BirthDate</th>
<th>Dim_Customer_FirstName</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5359696157</td>
<td>0.395891868</td>
</tr>
<tr>
<td>Dim_Customer_Gender</td>
<td>Dim_Customer_LastName</td>
</tr>
<tr>
<td>0.0668007149</td>
<td>0.0002234138</td>
</tr>
<tr>
<td>Dim_Customer_NameStyle</td>
<td>Dim_Customer_MiddleName</td>
</tr>
<tr>
<td>0.0002234138</td>
<td>0.0004468275</td>
</tr>
</tbody>
</table>
```

<table>
<thead>
<tr>
<th>fieldnames</th>
<th>x</th>
<th>field datatype</th>
<th>avg field</th>
<th>avg fieldnames</th>
<th>max field</th>
<th>max fieldnames</th>
<th>min field</th>
<th>min fieldnames</th>
<th>mapped_flg</th>
<th>train</th>
</tr>
</thead>
<tbody>
<tr>
<td>4457</td>
<td>G</td>
<td>character</td>
<td>0.9992923</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>DIM_CUSTOMER_Gender</td>
<td>1</td>
</tr>
<tr>
<td>4458</td>
<td>M</td>
<td>character</td>
<td>0.9992923</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>DIM_CUSTOMER_Gender</td>
<td>0</td>
</tr>
<tr>
<td>4459</td>
<td>F</td>
<td>character</td>
<td>0.9992923</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>DIM_CUSTOMER_Gender</td>
<td>1</td>
</tr>
<tr>
<td>4460</td>
<td>M</td>
<td>character</td>
<td>0.9992923</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>DIM_CUSTOMER_Gender</td>
<td>0</td>
</tr>
<tr>
<td>4461</td>
<td>F</td>
<td>character</td>
<td>0.9992923</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>DIM_CUSTOMER_Gender</td>
<td>1</td>
</tr>
<tr>
<td>4462</td>
<td>M</td>
<td>character</td>
<td>0.9992923</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>DIM_CUSTOMER_Gender</td>
<td>1</td>
</tr>
<tr>
<td>4463</td>
<td>F</td>
<td>character</td>
<td>0.9992923</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>DIM_CUSTOMER_Gender</td>
<td>1</td>
</tr>
<tr>
<td>4464</td>
<td>F</td>
<td>character</td>
<td>0.9992923</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>DIM_CUSTOMER_Gender</td>
<td>1</td>
</tr>
<tr>
<td>4465</td>
<td>F</td>
<td>character</td>
<td>0.9992923</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>DIM_CUSTOMER_Gender</td>
<td>1</td>
</tr>
<tr>
<td>4466</td>
<td>M</td>
<td>character</td>
<td>0.9992923</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>DIM_CUSTOMER_Gender</td>
<td>1</td>
</tr>
<tr>
<td>4467</td>
<td>M</td>
<td>character</td>
<td>0.9992923</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>DIM_CUSTOMER_Gender</td>
<td>0</td>
</tr>
<tr>
<td>4468</td>
<td>M</td>
<td>character</td>
<td>0.9992923</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>DIM_CUSTOMER_Gender</td>
<td>1</td>
</tr>
<tr>
<td>4469</td>
<td>F</td>
<td>character</td>
<td>0.9992923</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>DIM_CUSTOMER_Gender</td>
<td>0</td>
</tr>
<tr>
<td>4470</td>
<td>F</td>
<td>character</td>
<td>0.9992923</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>DIM_CUSTOMER_Gender</td>
<td>1</td>
</tr>
<tr>
<td>4471</td>
<td>NAME</td>
<td>character</td>
<td>0.9992923</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>DIM_CUSTOMER_Gender</td>
<td>1</td>
</tr>
<tr>
<td>4472</td>
<td>NAME</td>
<td>character</td>
<td>0.9992923</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>DIM_CUSTOMER_Gender</td>
<td>0</td>
</tr>
<tr>
<td>4473</td>
<td>NAME</td>
<td>character</td>
<td>0.9992923</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>DIM_CUSTOMER_Gender</td>
<td>1</td>
</tr>
<tr>
<td>4474</td>
<td>NAME</td>
<td>character</td>
<td>0.9992923</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>DIM_CUSTOMER_Gender</td>
<td>1</td>
</tr>
<tr>
<td>4475</td>
<td>NAME</td>
<td>character</td>
<td>0.9992923</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>DIM_CUSTOMER_Gender</td>
<td>1</td>
</tr>
<tr>
<td>4476</td>
<td>NAME</td>
<td>character</td>
<td>0.9992923</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>DIM_CUSTOMER_Gender</td>
<td>1</td>
</tr>
</tbody>
</table>
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 ---

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>492</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>367</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>62</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>f</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>g</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>h</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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.