PREDICTING INTERVIEW APPEARANCES

Document sign date :Jul 30, 2018

FORM NO.	NAME	COLLEGE NAME	UNIV ROLL NO.
39944	ANIRUDDHA SADHUKHAN	RCCIIT	11700115009
39943	SUDIPTA SARKAR	RCCIIT	11700115084
38628	SRIPARNA SARKAR	DIATM	15500115084
38652	ANASUA BAKSI	DIATM	15500115006
38653	SWARNALI DATTA	DIATM	15500115090

Document sign date :Jul 30, 2018

Table of Contents

ACKNOWLEDGEMENT

We take this opportunity to express our profound gratitude and deep regards to our faculty Mr. Titas Roy Chowdhury for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry us a long way in the journey of life on which we are about to embark.

We are obliged to our fellow project team members, for the valuable information provided by them in their respective fields. I am grateful to them for her cooperation during the period of the assignment.

Aniruddha Sadhukhan

Sudipta Sarkar

Sriparna Sarkar

Anasua Baksi

Swarnali Datta

PROJECT OBJECTIVE

We were given a dataset including various fields like client name, gender, location, answers to various questions asked over phone etc.

Based on this we have to predict the real appearance of candidates in the interview.

This is used by the recruiting agencies to predict if any possible candidates would really be present for the interview after talking over the phone.

SOURCE AND DESCRIPTION OF DATA

- The data has been provided by the institute
- The data set consists of 23 discrete specification columns and 1234 number of records

- **THE DIFFERENT SPECIFICATION CRITERIA(COLUMN) ARE AS FOLLOWS:-**
  - DATE OF INTERVIEW (COLUMN 1)-The date on which the interview has been scheduled. As per the given data set the date of interview does not have any effect on the observed attendance. Hence, we are not taking into consideration the date of interview for predicting the outcome.
  - CLIENT NAME (COLUMN 2)-The name of the company where the interview is scheduled.

    Unique values of client name-

    'Hospira', 'Aon Hewitt', 'UST', 'Standard Chartered Bank', 'ANZ', 'Pfizer', 'Standard Chartered Bank Chennai', 'Aon hewitt Gurgaon', 'Astrazeneca', 'Flextronics', 'Prodapt', 'Williams Lea', 'Barclays', 'Hewitt', 'Woori Bank'

  - INDUSTRY (COLUMN 3)-The type of industry where the interview has been fixed.

    Unique values of industry-

    'Pharmaceuticals', 'IT Services', 'BFSI', 'IT Products and Services', 'Electronics', 'Telecom', 'IT', NULL

  - LOCATION (COLUMN 4)-Location of the company.

    Unique values of location-

    'Chennai', 'Gurgaon', ' Bangalore', 'Hyderabad', 'Gurgaonr', 'Delhi', 'chennai', '- Cochin- ', 'Noida', 'CHENNAI', 'chennai ', NULL

- POSITION TO BE CLOSED (COLUMN 5)-The position that has been offered to the candidate.

  Unique values of position to be closed-'Production-Sterile' 'Selenium testing' 'Dot Net' 'AML' 'Trade Finance'
  'Routine' 'Niche' nan

- NATURE OF SKILLSET (COLUMN 6)-The type of skill required for the job.

  Unique values of nature of skillset-
  'Routine ', 'Oracle ', 'Accounting Operations ', 'Banking Operations ', 'Fresher ', 'AML/KYC/CDD ', 'CDD KYC ', 'Biosimiliars ', 'RA Label ', 'RA Publishing ', 'EMEA ', 'LCM -Manager ', 'Licensing â€" RA ', 'generic drugs â€" RA ', 'Biosimilars ', 'Regulatory ', 'Analytical R & D ', 'Analytical R&D ', 'Senior software engineer-Mednet ', 'Tech lead-Mednet ', 'Tech Lead- Mednet ', 'Technical Lead ', 'Sr Automation Testing ', 'TL ', 'Senior Analyst ', 'production ', 'Production ', 'Core Java ', 'Java J2EE ', 'Oracle Plsql ', 'Java,SQL ', 'Automation Testing Java ', 'Submission Management ', 'Biosimillar ', 'Publishing ', 'Global Labelling ', 'ALS Testing ', 'Java Developer ', 'Lending and Liabilities ', 'Lending & Liability ', 'Lending And Liabilities ', 'L & L ', 'Banking operations ', 'Lending&Liablities ', 'JAVA/J2EE/Struts/Hibernate ', 'JAVA/SPRING/HIBERNATE/JSF ', 'Java ', 'Java JSF ', 'Java,J2ee, JSF ', 'Java ,J2ee ', 'Java J2ee ', '11.30 AM ', '10.00 AM ', '9.00 Am ', '12.30 Pm ', '9.30 AM ', '11.30 Am ', 'Java, J2Ee ', 'Java,J2EE ', 'Java/J2ee/Core Java ', 'Java/J2ee ', 'JAVA, J2ee ', 'JAVA,J2ee ', 'T-24 developer ', 'COTS

```
Developer ', 'Product Control ', 'Dot Net ', 'COTS
', 'testing ', '- SAPBO, Informatica ', 'ETL ',
'Java-SAS ', 'Java Tech Lead ', 'Manager ',
'JAVA,SQL ', 'Java, SQL ', 'Hadoop ', 'SCCM ',
'SCCM-(Network, sharepoint,ms exchange) ', 'SCCm-
Desktop support ', 'Sccm- networking ', 'sccm ',
'SCCM â€" SQL ', 'SCCM â€" Sharepoint ',
'Production Support - SCCM ', 'SAS ', 'BaseSAS
Program/ Reporting ', 'Java, Spring, Hibernate ',
'Java,spring,hibernate ', 'Java, XML, Struts,
hibernate ', 'JAVA/J2EE ', 'Java ', NULL
```

- INTERVIEW TYPE (COLUMN 7)-Type of interview to be conducted.

  Unique values of interview type-

```
'Scheduled Walkin ', 'Scheduled  ', 'Walkin ',
'Scheduled Walk In ', 'Sceduledwalkin ', 'Walkin ',
NULL
```

- NAME (CAND ID) (COLUMN 8) –The name of the candidate. As per the given data set the candidate name and ID does not have any effect on the observed attendance. Hence, we are not taking this into consideration for predicting the outcome.

- GENDER (COLUMN 9)-The gender of the candidate.

  Unique values of gender-

```
'Male', 'Female', NULL
```

- CANDIDATE CURRENT LOCATION (COLUMN 10)- Current location of the candidate.

  If the current location of the candidate is same as that of the job location the candidate is likely to take up the interview.

  Unique values of candidate current location-

```
'Chennai ', 'Gurgaon ', 'Bangalore ', 'Hyderabad ',
'Delhi ', 'chennai ', '- Cochin-  ', 'Noida ',
'CHENNAI ', 'chennai ', NULL
```

- CANDIDATE JOB LOCATION (COLUMN 11)-Location of where the job is to be carried out.

  If the native location is same as the job location the candidate is likely to take up the job.

  Unique values of candidate job location-

```
'Hosur ', 'Bangalore ', 'Chennai ', 'Gurgaon ',
'Visakapatinam ', '- Cochin-  ', 'Noida', NULL
```

- INTERVIEW VENUE (COLUMN 12)-Venue where the interview is scheduled.

  If the interview venue is same as job location the candidate is likely to appear for the interview.

  Unique values of interview venue-

```
'Hosur ', 'Gurgaon ', 'Bangalore ', 'Chennai ',
'Hyderabad ', '- Cochin-  ', 'Noida', NULL
```

- CANDIDATE NATIVE LOCATION (COLUMN 13)-Native location of candidate.

  If the native location is same as the job location the candidate is likely to take up the job.

  Unique values of native location-

  'Hosur ', 'Trichy ', 'Chennai ', 'Gurgaon ', 'Noida ', 'Delhi /NCR ', 'Cochin' 'Trivandrum ', 'Bangalore ', 'Coimbatore ', 'Salem ', 'Tanjore ', 'Hyderabad' 'Mumbai ', 'Pune ', 'Kolkata ', 'Allahabad ', 'Panjim ', 'Cuttack ', 'Visakapatinam' 'Belgaum ', 'Patna ', 'Chitoor ', 'Anantapur ', 'Warangal ', 'Ahmedabad ', 'Kurnool' 'Vijayawada ', 'Vellore ', 'Pondicherry ', 'Nagercoil ', 'Agra ', 'Bhubaneshwar' 'Ghaziabad ', 'Baddi ', 'Tuticorin ', 'Tirupati ', 'Faizabad ', 'Ambur' 'Chandigarh ', 'Mysore ', 'Hissar ', 'Delhi ', 'Kanpur ', 'Lucknow ', '- Cochin- ', NULL

- HAVE YOU OBTAINED NECESSARY PERMISSION TO START AT THE REQUIRED TIME (COLUMN 14)-Set of questions that affect the appearance of interview.

  Unique values of this are –

  'Yes ', 'No ', 'Not yet ', 'Yet to confirm ', 'NO ', 'yes ', 'Na', NULL

- HOPE THERE WILL BE NO UNSCHEDULED MEETINGS TIME (COLUMN 15)-

  Unique values of this are-

  'Yes ', 'Na ', 'No ', 'yes ', 'Not Sure ', 'cant Say ', 'Not sure', NULL

- CAN I CALL YOU THREE HOURS BEFORE THE INTERVIEW AND FOLLOW AND FOLLOW UP ON YOUR ATTENDANCE FOR THE INTERVIEW (COLUMN 16) –Specifies the time when to make a call.

  Unique values of this are-

  `'Yes ', 'No ', 'No Dont ', 'Na ', 'yes', NULL`

  CAN I HAVE AN ALTERNATIVE NUMBER/DESK NUMBER(COLUMN 17)-

  Unique values –

  `'Yes ', 'No ', 'No I have only thi number ', 'na ', 'yes ', 'Na', NULL`

- ARE YOU CLEAR WITH THE VENUE DETAILS AND THE LANDMARK (COLUMN 19)-

  Unique values –

  `'Yes ', 'No ', 'No- I need to check ', 'na ', 'yes ', 'Na ', 'no', NULL`

- HAS THE CALL LETTER BEEN SHARED (COLUMN 20)-

  Unique values –

  `'Yes ', 'Havent Checked ', 'No ', 'Need To Check ', 'Not sure ', 'Yet to Check', 'Not Sure ', 'Not yet ', 'no ', 'na ', 'yes ', 'Na', NULL`

Document sign date :Jul 30, 2018

- EXPECTED ATTENDANCE (COLUMN 21)-

  Unique values –

  'Yes ', 'Uncertain ', 'No ', 'NO ', 'yes ', '11:00 AM ', '10.30 Am', NULL

- OBSERVED ATTENDANCE (COLUMN 22)-

  Unique values –

  'No ', 'Yes ', 'yes ', 'no ', 'yes ', 'No ', 'NO ', 'no ', NULL

- MARITAL STATUS (COLUMN 23)-

  Unique values –

  'Single', 'Married'

Document sign date :Jul 30, 2018

MODE CALCULATION OF GIVEN DATA

Client Name	Occurrence
Standard Chartered Bank	*904*
Hospira	*75*
Pfizer	*75*
Aon Hewitt	*28*
Flextronics	*23*
ANZ	22
Hewitt	20
UST	18
Standard Chartered Bank Chennai	17
Prodapt	17
Astrazeneca	15
Williams Lea	11
Barclays	5
Aon hewitt Gurgaon	2
Woori Bank	1
NULL	1

Therefore mode is:-904 (Standard Chartered Bank)

Industry	Occurrence
BFSI	949
Pharmaceuticals	165
IT Products and Services	45
Electronics	23
IT Services	23
Telecom	17
IT	11

Therefore mode is: 949 (BFSI)

Location	Occurrence
Chennai	754
Bangalore	292
chennai	86
Hyderabad	38
Gurgaon	33

Globsyn Knowledge Park Kolkata

Document sign date :Jul 30, 2018

Noida	15
- Cochin-	9
chennai	3
Delhi	1
Gurgaonr	1
CHENNAI	1

Therefore mode is = 754 (Chennai)

Position to be closed	Occurrence
Routine	1023
Niche	163
Dot Net	18
Trade Finance	11
AML	8
Selenium testing	5
Production- Sterile	5

Therefore mode is = 1023(routine)

Nature of skillset	Occurrence
JAVA/J2EE/Struts/Hibernate	220
Accounting Operations	86
Fresher	86
AML/KYC/CDD	84
CDD KYC	52
Routine	47
Oracle	43
JAVA/SPRING/HIBERNATE/JSF	42
Java J2EE	33
SAS	27
Oracle Plsql	25
Java Developer	25
Banking Operations	22
Lending and Liabilities	22
Java	21
Core Java	17
Java J2ee	16
T-24 developer	15
ALS Testing	15
Senior software engineer-Mednet	15
SCCM	14
COTS Develop--	13

Sr Automation Testing	13
Analytical R & D	13
Hadoop	12
Regulatory	12
testing	11
Java	10
Dot Net	9
Publishing	9
	...
TL	3
Biosimillar	3
11.30 AM	2
Java, Spring, Hibernate	2
Lending&Liablities	2
Banking operations	2
L & L	2
Production Support - SCCM	2
Submission Management	2
LCM -Manager	2
RA Label	2
11.30 Am	1
Technical Lead	1
SCCM SQL	1
SCCM Sharepoint	1
12.30 Pm	1
10.00 AM	1
9.30 AM	1
Tech Lead- Mednet	1
BaseSAS Program/ Reporting	1
sccm	1
Java, J2Ee	1
SCCM-(Network, sharepoint,ms exchange)	1
Production	1
Manager	1
Biosimilars	1
JAVA,J2ee	1
JAVA, J2ee	1
9.00 Am	1
Sccm- networking	1

Therefore mode is = 220(JAVA/J2EE/Struts/Hibernate)

Document sign date :Jul 30, 2018

Interview type	Occurrence
Scheduled Walk In	456
Scheduled	371
Walkin	189
Scheduled Walkin	189
Walkin	27
Sceduledwalkin	1

Therefore mode is = 456 (Scheduled Walk In)

Gender	Occurrence
Male	965
Female	268

Therefore mode is= 965(male)

Candidate current location	Occurrence
Chennai	*754*
Bangalore	*292*
chennai	*86*
Hyderabad	*38*
Gurgaon	34
Noida	15
- Cochin-	9
chennai	3
Delhi	1
CHENNAI	1

Therefore mode is = Chennai (754)

Candidate job location	Occurrence
Chennai	893
Bangalore	259
Gurgaon	35
Visakapatina	21

Document sign date :Jul 30, 2018

Noida	15
- Cochin-	9
Hosur	1

Therefore mode is = 893 (Chennai)

Interview venue	Occurrence
Chennai	852
Bangalore	277
Hyderabad	40
Gurgaon	35
Noida	15
- Cochin-	9
Hosur	5

Therefore mode is = 852 (Chennai)

Candidate Native location	Occurrence
Chennai	595
Hyderabad	172
Bangalore	151
Gurgaon	26
Cuttack	25
Cochin	24
Pune	22
Coimbatore	21
Allahabad	20
Noida	17
Nagercoil	16
Visakapatinam	16
Kolkata	14
Trivandrum	14
Trichy	13
Vellore	12
Mumbai	7
Chitoor	6
Pondicherry	5
Ahmedabad	5
- Cochin-	5
Chandigarh	5
Vijayawada	4

Tirupati	4
Delhi	4
Salem	3
Hosur	3
Warangal	3
Delhi /NCR	2
Ambur	2
Patna	2
Bhubaneshwar	1
Hissar	1
Tuticorin	1
Tanjore	1
Baddi	1
Agra	1
Faizabad	1
Kurnool	1
Mysore	1
Lucknow	1
Anantapur	1
Ghaziabad	1
Belgaum	1
Kanpur	1
Panjim	1

Therefore mode is = 595 (Chennai)

Have you obtained the necessary permission to start at the required time	Occurrence
Yes	917
No	79
Not yet	19
Na	5
yes	4
Yet to confirm	4
NO	1

Therefore mode is : 917 (yes)

Hope there will be no unscheduled meetings	Occurrence
Yes	949
Na	20
No	6
yes	5

Not sure	4
cant Say	1
Not Sure	1

Therefore mode is = 949 (Yes)

Can I Call you three hours before the interview and follow up on your attendance for the interview	Occurrence
Yes	951
Na	20
No	10
yes	4
No Dont	1

Therefore mode is = 951 (Yes)

Can I have an alternative number/ desk number. I assure you that I will not trouble you too much	Occurrence
Yes	936
No	27
Na	19
No I have only thi number	2
na	1
yes	1

Therefore mode is = 936 (mode)

Have you taken a printout of your updated resume. Have you read the JD and understood the same	Occurrence
Yes	940
Na	19
No	16
Not Yet	4
yes	2
Not yet	2

na	1
No- will take it soon	1

Therefore mode is = 940 (Yes)

Are you clear with the venue details and the Landmark	Occurrence
Yes	946
Na	19
No	14
yes	2
No- I need to check	2
na	1
no	1

Therefore mode is =946 (Yes)

Has the call Letter Been shared	Occurrence
Yes	932
Na	19
No	17
Not Sure	8
Need To Check	3
yes	2
Not yet	2
na	1
Not sure	1
Havent Checked	1
Yet to Check	1
no	1

Therefore mode is = 932 (Yes)

Expected Attendance	Occurrence
Yes	882
Uncertain	250
No	59
NO	34
yes	1
11:00 AM	1
10.30 Am	1

Therefore mode is = 882(yes)

Observed attendance	Occurrence
Yes	701
No	401
yes	81
NO	35
no	7
No	6
yes	1
no	1

Therefore mode is = 701 (yes)

TECHNIQUES USED

➢ *Supervised Machine Learning*

The majority of practical machine learning uses supervised learning.

Supervised learning is where we have input variables (x) and an output variable (Y) and we use an algorithm to learn the mapping function from the input to the output.

$$Y = f(X)$$

The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.

It is called supervised learning because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers, the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.

➢ *Binary classification*

Binary or binomial classification is the task of classifying the elements of a given set into two groups (predicting which group each one belongs to) on the basis of a classification rule.

The actual output of many binary classification algorithms is a prediction score. The score indicates the system's certainty that the given observation belongs to the positive class.

Binary Classification would generally fall into the domain of **Supervised Learning** since the training dataset is labelled. And as the name suggests it is simply a special case in which there are only two classes.

➢ ***Naive Bayes classifiers***

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes' Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

To start with, let us consider a dataset.

Consider a fictional dataset that describes the weather conditions for playing a game of golf. Given the weather conditions, each tuple classifies the conditions as fit("Yes") or unfit("No") for playing golf.

Here is a tabular representation of our dataset.

	Outlook	Temperature	Humidity	Windy	Play Golf
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

The dataset is divided into two parts, namely, **feature matrix** and the **response vector**.

- Feature matrix contains all the vectors(rows) of dataset in which each vector consists of the value of **dependent features**. In above dataset, features are 'Outlook', 'Temperature', 'Humidity' and 'Windy'.
- Response vector contains the value of **class variable**(prediction or output) for each row of feature matrix. In above dataset, the class variable name is 'Play golf'.

Assumption:

The fundamental Naive Bayes assumption is that each feature makes an:

- independent
- equal

contribution to the outcome.

With relation to our dataset, this concept can be understood as:

- We assume that no pair of features are dependent. For example, the temperature being 'Hot' has nothing to do with the humidity or the outlook being 'Rainy' has no effect on the winds. Hence, the features are assumed to be **independent**.
- Secondly, each feature is given the same weight(or importance). For example, knowing only temperature and humidity alone can't predict the outcome accurately. None of the attributes is irrelevant and assumed to be contributing **equally** to the outcome.

Note: The assumptions made by Naive Bayes are not generally correct in real-world situations. In-fact, the independence assumption is never correct but often works well in practice.

Now, before moving to the formula for Naive Bayes, it is important to know about Bayes' theorem.

Bayes' Theorem

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

where A and B are events and P(B) ≠ 0.

- Basically, we are trying to find probability of event A, given the event B is true. Event B is also termed as **evidence**.
- P(A) is the **priori** of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance(here, it is event B).
- P(A|B) is a posteriori probability of B, i.e. probability of event after evidence is seen.

Now, with regards to our dataset, we can apply Bayes' theorem in following way:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

where, y is class variable and X is a dependent feature vector (of size *n*) where:

$$X = (x_1, x_2, x_3, ..., x_n)$$

Just to clear, an example of a feature vector and corresponding class variable can be: (refer 1st row of dataset)

X = (Rainy, Hot, High, False)
y = No

So basically, P(X|y) here means, the probability of "Not playing golf" given that the weather conditions are "Rainy outlook", "Temperature is hot", "high humidity" and "no wind".

Naive assumption

Now, it's time to put a naive assumption to the Bayes' theorem, which is, **independence** among the features. So now, we split **evidence** into the independent parts.

Now, if any two events A and B are independent, then,

P(A,B) = P(A)P(B)

Hence, we reach to the result:

$$P(y|x_1,...,x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

which can be expressed as:

$$P(y|x_1,...,x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1)P(x_2)...P(x_n)}$$

Now, as the denominator remains constant for a given input, we can remove that term:

$$P(y|x_1,...,x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y)$$

Now, we need to create a classifier model. For this, we find the probability of given set of inputs for all possible values of the class variable *y* and pick up the output with maximum probability. This can be expressed mathematically as:

$$y = argmax_y P(y)\prod_{i=1}^{n} P(x_i|y)$$

So, finally, we are left with the task of calculating P(y) and $P(x_i \mid y)$.

Please note that P(y) is also called **class probability** and $P(x_i \mid y)$ is called **conditional probability**.

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$

Let us try to apply the above formula manually on our weather dataset. For this, we need to do some precomputations on our dataset.

We need to find $P(x_i \mid y_j)$ for each x_i in X and y_j in y. All these calculations have been demonstrated in the tables below:

Outlook

	Yes	No	P(yes)	P(no)
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rainy	3	2	3/9	2/5
Total	9	5	100%	100%

Temperature

	Yes	No	P(yes)	P(no)
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
Total	9	5	100%	100%

Humidity

	Yes	No	P(yes)	P(no)
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
Total	9	5	100%	100%

Wind

	Yes	No	P(yes)	P(no)
False	6	2	6/9	2/5
True	3	3	3/9	3/5
Total	9	5	100%	100%

Play		P(Yes)/P(No)
Yes	9	9/14
No	5	5/14
Total	14	100%

So, in the figure above, we have calculated $P(x_i \mid y_j)$ for each x_i in X and y_j in y manually in the tables 1-4. For example, probability of playing golf given that the temperature is cool, i.e P(temp. = cool | play golf = Yes) = 3/9.

Also, we need to find class probabilities (P(y)) which has been calculated in the table 5. For example, P(play golf = Yes) = 9/14.

So now, we are done with our pre-computations and the classifier is ready!

Let us test it on a new set of features (let us call it today):

today = (Sunny, Hot, Normal, False)

So, probability of playing golf is given by:

$$P(Yes|today) = \frac{P(SunnyOutlook|Yes)P(HotTemperature|Yes)P(NormalHumidity|Yes)P(NoWind|Yes)P(Yes)}{P(today)}$$

and probability to not play golf is given by:

$$P(No|today) = \frac{P(SunnyOutlook|No)P(HotTemperature|No)P(NormalHumidity|No)P(NoWind|No)P(No)}{P(today)}$$

Since, P(today) is common in both probabilities, we can ignore P(today) and find proportional probabilities as:

$$P(Yes|today) \propto \frac{2}{9}.\frac{2}{9}.\frac{6}{9}.\frac{6}{9}.\frac{9}{14} \approx 0.0141$$

and

$$P(No|today) \propto \frac{3}{5}.\frac{2}{5}.\frac{1}{5}.\frac{2}{5}.\frac{5}{14} \approx 0.0068$$

Now, since

$$P(Yes|today) + P(No|today) = 1$$

These numbers can be converted into a probability by making the sum equal to 1 (normalization):

$$P(Yes|today) = \frac{0.0141}{0.0141+0.0068} = 0.67$$

and

$$P(No|today) = \frac{0.0068}{0.0141+0.0068} = 0.33$$

Since

$$P(Yes|today) > P(No|today)$$

So, prediction that golf would be played is ‘Yes’.

The method that we discussed above is applicable for discrete data. In case of continuous data, we need to make some assumptions regarding the distribution of values of each feature. The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$.

HARDWARE AND SOFTWARE REQUIREMENTS

HADOOP CLUSTER IS USED WHICH IS MADE OF COMMODITY HARDWARE

➢ *MASTER'S RECOMMENDATION*

- ❖ 4–6 1TB hard disks
  (1 for OS [RAID 1], 2 for the FS image [RAID 5/6], 1 for JournalNode)
- ❖ 2 CPUs(8-12 cores per CPU), running at least 2-2.5GHz
- ❖ 128-512GB of RAM
- ❖ Bonded Gigabit Ethernet or 10Gigabit Ethernet

➢ *SLAVE'S RECOMMENDATION*

- ❖ 12-24 1-4TB hard disks in a JBOD (Just a Bunch Of Disks) configuration
- ❖ 2 CPUs(8-12 cores per CPU), running at least 2-2.5GHz
- ❖ 64-128GB of RAM
- ❖ Bonded Gigabit Ethernet or 10Gigabit Ethernet

➢ *SOFTWARE*

- ❖ Hadoop 2.2.0 or above runs well on Linux operating systems like: RedHat Enterprise Linux (RHEL), CentOS, Ubuntu
- ❖ Hadoop is written in Java. The recommended Java version is Oracle JDK 1.6.31

DATA CLEANSING

- Rows in which some data were missing or all the 23 data weren't present are ignored
- Some data included incorrect punctuation marks or non-word characters and are rectified
- Some values were nearly similar(e.g: liaison and liabilities, liaison & liabilities, liaison liabilities)and were changed to a single value
- Rows where some discrepancy in data was there were rectified

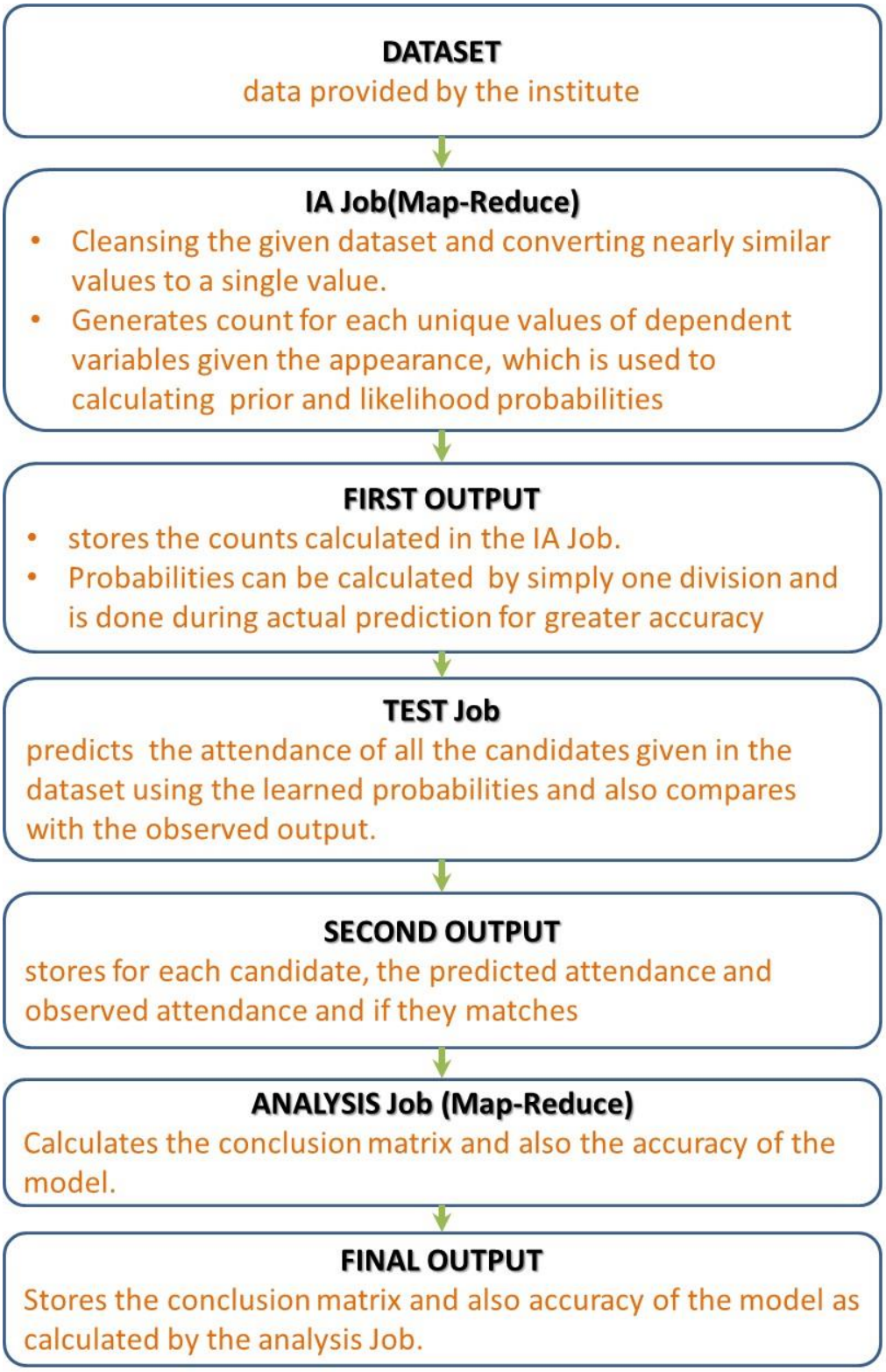
DATA PROCESSING

Concept of Map-Reduce has been used to process the data.

Three jobs have been used namely:-

- ***IA Job*** : *MAPPER with multiple REDUCERS (depending on size of dataset)*
- ***TEST Job****: Only MAPPER and no REDUCER*
- ***ANALYSIS Job****: MAPPER with 1 REDUCER*

WORK FLOW DIAGRAMS



Document sign date :Jul 30, 2018

CODE

Code for package ani

IAMapper.java

```
package ani;

import java.io.IOException;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.commons.lang.StringUtils;

public class IAMapper extends Mapper<LongWritable,Text,Text,TupleWritable>
{
	private Text outkey = new Text();
	private TupleWritable outval = new TupleWritable();
	private int i;

	@Override
	protected void map(LongWritable key, Text value,Context context)
	throws IOException, InterruptedException
	{
		if(key.get()!=0)
		{
			String line = value.toString().trim();
			String word[] = line.split(",");

			if(word!=null && word.length >22 && word[7]!=null)
			{
				try
				{
					for(i=1;i<20;i++)
					{
						if(i>0 && i<5)
						{

outkey.set("x"+i+"_"+word[i].toLowerCase().replaceAll("[^A-Za-z]+", ""));
						}
						else if(i==5)
						{
```

```
                                        if(Character.isDigit(word[5].charAt(0)))
                                                outkey.set("x5_unknown");
                                        else{
                                                word[5] =
word[5].toLowerCase().replaceAll("[^A-Za-z]+","");

        if(StringUtils.getLevenshteinDistance(word[5] ,"biosimilars")<4)
                                                        outkey.set("x5_biosimilars");
                                                else
if(StringUtils.getLevenshteinDistance(word[5] ,"lendingliablities")<11)

        outkey.set("x5_lendingliablities");
                                                else outkey.set("x5_"+word[5]);
                                        }
                                }
                                else if(i==6)
                                {
                                        word[6] =
word[6].toLowerCase().replaceAll("[^A-Za-z]+","");

        if(StringUtils.getLevenshteinDistance(word[6] ,"scheduledwalkin")<3)
                                                outkey.set("x6_scheduledwalkin");
                                        else outkey.set("x6_"+word[6]);
                                }
                                else if(i==7)
                                {
                                        if(word[8].equalsIgnoreCase("male"))
                                                outkey.set("x7_male");
                                        else outkey.set("x7_female");
                                }
                                else if(i==8)
                                {
                                        if(word[9].equalsIgnoreCase(word[10]))
                                                outkey.set("x8_yes");
                                        else outkey.set("x8_no");
                                }
                                else if(i==9)
                                {
                                        if(word[9].equalsIgnoreCase(word[11]))
                                                outkey.set("x9_yes");
                                        else outkey.set("x9_no");
                                }
                                else if(i==10)
                                {
```

```
                    if(word[12].equalsIgnoreCase(word[10]))
                        outkey.set("x10_yes");
                    else outkey.set("x10_no");
                }
                else if(i>10 && i<14)
                {
                    if(word[i+2].equalsIgnoreCase("yes"))
                        outkey.set("x"+i+"_"+"yes");
                    else if(word[i+2].equalsIgnoreCase("no"))
                        outkey.set("x"+i+"_"+"no");
                    else if(word[i+2].equalsIgnoreCase("NA"))
                        outkey.set("x"+i+"_"+"na");
                    else
                        outkey.set("x"+i+"_"+"uncertain");
                }
                else if(i>13 && i<17)
                {
                    if(word[i+3].equalsIgnoreCase("yes"))
                        outkey.set("x"+i+"_"+"yes");
                    else if(word[i+3].equalsIgnoreCase("no"))
                        outkey.set("x"+i+"_"+"no");
                    else if(word[i+3].equalsIgnoreCase("NA"))
                        outkey.set("x"+i+"_"+"na");
                    else
                        outkey.set("x"+i+"_"+"uncertain");
                }
                else if(i==17)
                {
                    if(word[20].equalsIgnoreCase("no"))
                        outkey.set("x17_"+"no");
                    else
if(word[20].equalsIgnoreCase("UNCERTAIN"))
                        outkey.set("x17_"+"uncertain");
                    else if(word[20].equalsIgnoreCase("NA"))
                        outkey.set("x17_"+"na");
                    else
                        outkey.set("x17_"+"yes");
                }
                else if(i==18)
                {
                    if(word[22].equalsIgnoreCase("Single"))
                        outkey.set("x18_s");
                    else outkey.set("x18_m");
                }
```

```
                                        else if(i==19)
                                        {
                                                outkey.set("y");
                                        }



                                        if(word[21].equalsIgnoreCase("yes"))
                                                outval.set(1, 0);
                                        else outval.set(0, 1);

                                        context.write(outkey, outval);
                                }

                        }catch(Exception e){}


                }

        }

}
```

IAReducer.java

```
package ani;

import java.io.IOException;
import java.util.Iterator;

import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Reducer;

public class IAReducer extends Reducer<Text, TupleWritable, Text, TupleWritable>
{
        private TupleWritable outval = new TupleWritable();
        @Override
        protected void reduce(Text key, Iterable<TupleWritable> values,Context context)
```

```
                throws IOException, InterruptedException
        {
                int sumy = 0,sumn = 0;
                for(TupleWritable t : values)
                {
                        sumy = sumy + t.getYes();
                        sumn = sumn + t.getNo();
                }
                outval.set(sumy,sumn);
                context.write(key, outval);
        }

}
```

TupleWritable.java

```
package ani;

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;

import org.apache.hadoop.io.*;

public class TupleWritable implements Writable
{
        private IntWritable yes = new IntWritable();
        private IntWritable no= new IntWritable();

        public void set(int y,int n)
        {
                yes.set(y);
                no.set(n);
        }

        public int getNo()
        {
                return no.get();
        }

        public int getYes()
        {
                return yes.get();
```

```
	}

	@Override
	public void readFields(DataInput inpstream) throws IOException
	{
			yes.readFields(inpstream);
			no.readFields(inpstream);
	}

	@Override
	public void write(DataOutput outstream) throws IOException
	{
			yes.write(outstream);//read write order to be maintained
			no.write(outstream);
	}

	@Override
	public String toString() {
		return String.valueOf(yes.get())+"\t"+String.valueOf(no.get());
	}

}
```

IADriver.java

```
package ani;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class IADriver
{
```

```
	public static void main(String[] args) throws IOException, ClassNotFoundException,
InterruptedException
	{
		Configuration config = new Configuration();
		Job job = Job.getInstance(config);

		job.setJarByClass(IADriver.class);
		job.setMapperClass(IAMapper.class);
		job.setReducerClass(IAReducer.class);
		job.setCombinerClass(IAReducer.class);

		job.setOutputKeyClass(Text.class);
		job.setOutputValueClass(TupleWritable.class);


		job.setNumReduceTasks(3);
		FileInputFormat.addInputPath(job, new Path("datasets"));//can be called
multiple times

		FileOutputFormat.setOutputPath(job, new Path("ProbabilitiesLists"));//output
path must be one
		//Output file if preexisted will not overwrite,throw Exception


		job.waitForCompletion(true);//starts processing

	}


}
```

Code for package test

TestMapper.java

```
package test;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;

import org.apache.commons.lang.StringUtils;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Mapper;

public class TestMapper extends Mapper<LongWritable, Text, Text, Text>
{
	HashMap<String,String> map = new HashMap<String,String>();
	private Text outkey = new Text(),outval = new Text();


	@Override
	protected void setup(Context context)
	throws IOException, InterruptedException
	{
		Path path[] = context.getLocalCacheFiles();
		if (path != null && path.length > 0)
		{
			for(Path p : path)
			{
				String strpath = p.toString();
				FileReader file = new FileReader(strpath);
				BufferedReader breader = new BufferedReader(file);

				while(true)
				{
					String line = breader.readLine();
					if (line == null) break;
					String words[] = line.split("\t");
					map.put(words[0],words[1]+" "+words[2]);
				}
```

```
                breader.close();
                file.close();
            }
        }
    }


    @Override
    protected void map(LongWritable key, Text value,Context context)
    throws IOException, InterruptedException
    {
        if(key.get()!=0)
        {
            String line = value.toString().trim();
            String word[] = line.split(","),attr ="",appear;
            int i;
            double Py,Pn, x[] = new double[2] ,ny[]= new double[2] ;
            if(word!=null && word.length >22 && word[7]!=null &&
word[7].startsWith("Candidate"))
            {
                ny[0] = Double.valueOf(map.get("y").split(" ")[0]);
                ny[1] = Double.valueOf(map.get("y").split(" ")[1]);

                //Initialise Py with P(y=yes) & Pn with P(y=no)
                Py = ny[0]/(ny[0]+ny[1]);
                Pn = ny[1]/(ny[0]+ny[1]);

                for(i=2;i<19;i++)
                {
                    if(i>0 && i<5)
                    {

    attr="x"+i+"_"+word[i].toLowerCase().replaceAll("[^A-Za-z]+", "");
                    }
                    else if(i==5)
                    {
                        attr = word[5].toLowerCase().replaceAll("[^A-Za-
z]+", "");
                        if(StringUtils.getLevenshteinDistance(attr
,"biosimilars")<4)
                            attr="x5_biosimilars";
                        else
if(StringUtils.getLevenshteinDistance(attr,"lendingliabilities")<11)
```

```
                        attr="x5_lendingliablities";
                    else attr="x5_"+attr;
                }
                else if(i==6)
                {
                    attr = word[6].toLowerCase().replaceAll("[^A-Za-
z]+", "");
                    if(StringUtils.getLevenshteinDistance(attr
,"scheduledwalkin")<3)
                        attr = "x6_scheduledwalkin";
                    else attr = "x6_"+attr;

                }
                else if(i==7)
                {
                    if(word[8].equalsIgnoreCase("male"))
                        attr = "x7_male";
                    else attr = "x7_female";
                }
                else if(i==8)
                {
                    if(word[9].equalsIgnoreCase(word[10]))
                        attr = "x8_yes";
                    else attr = "x8_no";
                }
                else if(i==9)
                {
                    if(word[9].equalsIgnoreCase(word[11]))
                        attr = "x9_yes";
                    else attr = "x9_no";
                }
                else if(i==10)
                {
                    if(word[12].equalsIgnoreCase(word[10]))
                        attr = "x10_yes";
                    else attr = "x10_no";
                }
                else if(i>10 && i<14)
                {
                    if(word[i+2].equalsIgnoreCase("yes"))
                        attr = "x"+i+"_"+"yes";
                    else if(word[i+2].equalsIgnoreCase("no"))
                        attr = "x"+i+"_"+"no";
                    else if(word[i+2].equalsIgnoreCase("NA"))
```

```
                    attr = "x"+i+"_"+"na";
                else
                    attr = "x"+i+"_"+"uncertain";
            }
            else if(i>13 && i<17)
            {
                if(word[i+3].equalsIgnoreCase("yes"))
                    attr = "x"+i+"_"+"yes";
                else if(word[i+3].equalsIgnoreCase("no"))
                    attr = "x"+i+"_"+"no";
                else if(word[i+3].equalsIgnoreCase("NA"))
                    attr = "x"+i+"_"+"na";
                else
                    attr = "x"+i+"_"+"uncertain";
            }
            else if(i==17)
            {
                if(word[20].equalsIgnoreCase("no"))
                    attr = "x17_"+"no";
                else if(word[20].equalsIgnoreCase("UNCERTAIN"))
                    attr = "x17_"+"uncertain";
                else if(word[20].equalsIgnoreCase("NA"))
                    attr = "x17_"+"na";
                else
                    attr = "x17_"+"yes";
            }
            else if(i==18)
            {
                if(word[22].equalsIgnoreCase("Single"))
                    attr = "x18_s";
                else attr = "x18_m";
            }

            if (map.containsKey(attr))
            {
                x[0] = Double.valueOf(map.get(attr).split(" ")[0]);
                x[1] = Double.valueOf(map.get(attr).split(" ")[1]);

                //Updating Py = Py * P(x=attr | y = yes)
                //Updating Pn = Pn * P(x=attr | y = yes)
                Py = Py * x[0] / ny[0];
                Pn = Pn * x[1] / ny[1];
            }
        }
```

```
                        //"yes = "+Py*100+"% \t no = "+Pn*100+"%"

                        if (Py>=Pn)
                                appear = "yes";
                        else appear = "no ";

                        if(appear.trim().equalsIgnoreCase(word[21].trim()))
                                outkey.set("Output Matched    ");
                        else outkey.set("Output Not Matched");
                        outval.set("Observed : "+word[21].trim()+"\tPredicted :
"+appear+" <--"+word[7].trim());

                        context.write(outkey, outval);
                    }
                }
        }
}
```

TestDriver.java

```
package test;

import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import ani.IADriver;
import ani.IAMapper;
import ani.IAReducer;
import ani.TupleWritable;
```

```
public class TestDriver
{
	public static void main(String[] args) throws IOException, ClassNotFoundException,
InterruptedException, URISyntaxException
	{
		Configuration config = new Configuration();
		Job job = Job.getInstance(config);

		job.addCacheFile(new URI("/user/edureka/ProbabilitiesLists/part-r-00000"));
		job.addCacheFile(new URI("/user/edureka/ProbabilitiesLists/part-r-00001"));
		job.addCacheFile(new URI("/user/edureka/ProbabilitiesLists/part-r-00002"));

		job.setJarByClass(TestDriver.class);
		job.setMapperClass(TestMapper.class);

		job.setOutputKeyClass(Text.class);
		job.setOutputValueClass(Text.class);


		job.setNumReduceTasks(1);
		FileInputFormat.addInputPath(job, new Path("datasets"));

		FileOutputFormat.setOutputPath(job, new Path("OutputMatches"));


		job.waitForCompletion(true);
	}

}
```

Code for package analysis

AnalysisMapper.java

```
package analysis;

import java.io.IOException;

import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Mapper;

public class AnalysisMapper extends Mapper<LongWritable, Text, NullWritable, Text>
{
	private Text outval = new Text();
	private NullWritable outkey = NullWritable.get();

	@Override
	protected void map(LongWritable key, Text value,Context context)
	throws IOException, InterruptedException
	{
			String line = value.toString().trim();
			String word[] = line.split("\t");
			String observed  = word[1].split(" ")[2].toLowerCase().trim();
			String predicted = word[2].split(" ")[2].toLowerCase().trim();

			outval.set(observed+" "+predicted);


			context.write(outkey, outval);

	}
}
```

Document sign date :Jul 30, 2018

AnalysisReducer.java

```
package analysis;

import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Reducer;

public class AnalysisReducer extends Reducer<NullWritable, Text, NullWritable, Text>
{
	private Text outval = new Text();
	@Override
	protected void reduce(NullWritable key, Iterable<Text> values,Context context)
			throws IOException, InterruptedException
	{
		Iterator<Text> itr = values.iterator();
		int yy= 0,yn = 0, ny = 0,nn = 0;
		float accuracy;
		while(itr.hasNext())
		{
			outval = itr.next();
			String observed  = outval.toString().split(" ")[0].trim();
			String predicted = outval.toString().split(" ")[1].trim();

			if(observed.equals("yes") && predicted.equals("yes"))	yy=yy+1;
			else if (observed.equals("yes") && predicted.equals("no"))  yn=yn+1;
			else if (observed.equals("no") && predicted.equals("yes"))  ny=ny+1;
			else if (observed.equals("no") && predicted.equals("no")) nn=nn+1;

		}
		accuracy = ((float)(yy+nn)/(yy+yn+ny+nn)*100);
		String out =	"Observed yes , Predicted yes : "+yy+
					"\nObserved yes , Predicted no  : "+yn+
					"\nObserved no  , Predicted yes : "+ny+
					"\nObserved no  , Predicted no  : "+nn+
					"\n\nAccuracy : "+accuracy+"%";

		outval.set(out);
		context.write(key, outval);
	}

}
```

AnalysisDriver.java

```
package analysis;

import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class AnalysisDriver
{
	public static void main(String[] args) throws IOException, ClassNotFoundException,
InterruptedException, URISyntaxException
	{
		Configuration config = new Configuration();
		Job job = Job.getInstance(config);

		job.setJarByClass(AnalysisDriver.class);
		job.setMapperClass(AnalysisMapper.class);
		job.setReducerClass(AnalysisReducer.class);

		job.setOutputKeyClass(NullWritable.class);
		job.setOutputValueClass(Text.class);

		job.setNumReduceTasks(1);
		FileInputFormat.addInputPath(job, new Path("OutputMatches"));
		FileOutputFormat.setOutputPath(job, new Path("Analysis"));


		job.waitForCompletion(true);
	}

}
```

OUTPUT

Output of IA job:

A small portion of the output of IA job is given below:

```
x11_no      6     73
x11_yes     651   240
x13_na      103   164
x14_no      2     14
x14_yes     682   260
x15_uncertain     1     1
x16_na      103   162
x17_no      1     92
x17_yes     666   219
x1_aonhewitt      24    4
x1_barclays       5     0
x1_pfizer   51    24
x1_prodapt  6     11
x1_ust      10    8
x2_itproductsandservices    34    11
x2_pharmaceuticals    96    69
x3_bangalore      193   99
x3_gurgaon  22    11
x4_dotnet   10    8
x4_productionsterile  0     5
x4_seleniumtesting    4     1
```

The first column states each unique values of the dependent variables and next two columns indicates the count of those unique values in the dataset given Observed Attendance = Yes and No respectively.

Document sign date :Jul 30, 2018

Output of Test Job:

A small portion of the output of Test job is given below:

```
Output Matched      Observed : No       Predicted : no  <--Candidate 1
Output Matched      Observed : Yes      Predicted : yes <--Candidate 1232
Output Matched      Observed : Yes      Predicted : yes <--Candidate 1231
Output Matched      Observed : Yes      Predicted : yes <--Candidate 1230
Output Matched      Observed : Yes      Predicted : yes <--Candidate 1229
Output Matched      Observed : Yes      Predicted : yes <--Candidate 1228
Output Matched      Observed : Yes      Predicted : yes <--Candidate 1227
Output Matched      Observed : Yes      Predicted : yes <--Candidate 1226
Output Matched      Observed : Yes      Predicted : yes <--Candidate 1225
Output Matched      Observed : Yes      Predicted : yes <--Candidate 1224
Output Matched      Observed : Yes      Predicted : yes <--Candidate 1223
Output Matched      Observed : Yes      Predicted : yes <--Candidate 1222
Output Matched      Observed : Yes      Predicted : yes <--Candidate 1221
Output Matched      Observed : Yes      Predicted : yes <--Candidate 1220
Output Matched      Observed : Yes      Predicted : yes <--Candidate 1219
Output Matched      Observed : Yes      Predicted : yes <--Candidate 1218
                                 :
                                 :
Output Not Matched     Observed : No    Predicted : yes <--Candidate 730
Output Not Matched     Observed : No    Predicted : yes <--Candidate 729
Output Not Matched     Observed : No    Predicted : yes <--Candidate 1025
Output Not Matched     Observed : No    Predicted : yes <--Candidate 728
Output Not Matched     Observed : No    Predicted : yes <--Candidate 204
Output Not Matched     Observed : No    Predicted : yes <--Candidate 525
Output Not Matched     Observed : No    Predicted : yes <--Candidate 279
Output Not Matched     Observed : No    Predicted : yes <--Candidate 1020
Output Not Matched     Observed : No    Predicted : yes <--Candidate 724
Output Not Matched     Observed : No    Predicted : yes <--Candidate 1018
Output Not Matched     Observed : No    Predicted : yes <--Candidate 1017
Output Not Matched     Observed : Yes   Predicted : no  <--Candidate 523
Output Not Matched     Observed : Yes   Predicted : no  <--Candidate 124
Output Not Matched     Observed : Yes   Predicted : no  <--Candidate 521
Output Not Matched     Observed : Yes   Predicted : no  <--Candidate 379
Output Not Matched     Observed : No    Predicted : yes <--Candidate 202
```

Document sign date :Jul 30, 2018

Output of Analysis Job (Final Output):

```
Observed yes , Predicted yes : 650
Observed yes , Predicted no  : 111
Observed no  , Predicted yes : 214
Observed no  , Predicted no  : 216

Accuracy : 72.712006%
```

This final output gives the Conclusion Matrix and also the accuracy of all the predictions made.

It is to be noted that the most important part of the Conclusion Matrix is when the algorithm Predicts No but it is Observed Yes. Because if the prediction matches, it's great but if it is predicted Yes and Observed No, then no big problem can arise. Problem occurs when it is predicted No and so no preparations are made for those candidates but the candidates comes to attend the interview. So we should keep this as low as possible.

FUTURE IMPROVEMENT

The prediction analysis has been done based on a batch processing system instead of a real time system. For further improvement this has to be transformed into real time system using Apache Spark.

OTHER DOMAINS WHERE THIS CONCEPTS CAN BE USED

This Naive Bayes approach using MapReduce can be used similarly in many other applications:

- Churn Detection
- Vote Prediction
- Email Spam Detection
- News article categorization
- Sentiment Analysis
- Facial recognition
- Handwriting Recognition
- Weather Prediction

CONCLUSION

From a given data set, we have calculated the conditional probability for each value of the dependent attributes given the appearance of the candidates using Naive Bayes supervised machine learning.

After that we have predicted the candidate appearance and cross checked with the observed appearance and we have got a satisfactory **73% accuracy**. So we can say that this application is working as expected.