Jan, 2018

TEXT - CLASSIFICATION

Ayyappa Bhuma

Contents

Problem Statement and Background:

XYZ Health Services is a top ranked Health care provider in USA with stellar credentials and provides high quality-care with focus on end-to-end Health care services. The Heath Care Services range from basic medical diagnostics to critical emergency services. The provider follows a ticketing system for all the telephonic calls received across all the departments. Calls to the provider can be for New Appointment, Cancellation, Lab Queries, Medical Refills, Insurance Related, and General Doctor Advise etc. The Tickets have the details of Summary of the call and description of the calls written by various staff members with no standard text guidelines.
The challenge is, based on the Text in the Summary and Description of the call, the ticket is to be classified to Appropriate Category (out of 5 Categories) and Subcategories (Out of 20 Sub Categories).

DATA:

fileid	SUMMARY	DATA	categories	sub_categories	previous_	ID
2.02E+12	Pt aware that h	{\rtf1\ansi\ftnbj{	PRESCRIPTION	REFILL	No	2015_5_6133_1001
2.02E+12	Mom wants to	{\rtf1\ansi\ftnbj{	ASK_A_DOCTOR	MEDICATION RELATED	No	2015_5_6134_1001
2.02E+12	pt called to dis	xxxx-xxxx\f0 \fsv	ASK_A_DOCTOR	MEDICATION RELATED	No	2015_5_6135_1001
2.02E+12	FYI Nortryptlin	xxxx-xxxx\f0 \fsv	MISCELLANEOUS	OTHERS	No	2015_5_6136_1001
2.02E+12	Letter of patie	{\rtf1\ansi\ftnbj{	MISCELLANEOUS	SHARING OF HEALTH RECORDS (FAX,	No	2015_5_6137_1001
2.02E+12	Appt question	{\rtf1\ansi\ftnbj{	APPOINTMENTS	QUERY ON CURRENT APPOINTMENT	No	2015_5_6140_1001
2.02E+12	dizzy & double	xxxx-xxxx\f0 \fsv	ASK_A_DOCTOR	SYMPTOMS	No	2015_5_6141_1001
2.02E+12	Please refax n	xxxx-xxxx\f0 \fsv	MISCELLANEOUS	SHARING OF HEALTH RECORDS (FAX,	No	2015_5_6142_1001
2.02E+12	pt wants to res	xxxx-xxxx\f0 \fsv	APPOINTMENTS	RESCHEDULING	No	2015_5_6144_1001

From problem statement it is evident that we need to build a classification model to predict 2 variables “categories” and “sub_categories”. Here, there are 5 independent variables: “fileid”, “summary”, “data”, “previous_appointment” and “id” using which we need to predict the target variables.

Domain Knowledge:

Before diving into the problem we need to understand the problem in its domain perspective rather than in data science perspective.

The problem is to classify the customer’s phone call into a category to which it belongs and then into a subcategory in order to classify it further.

As subcategory is based on category our problem solving should start from finding the category to which the customer’s phone call belong.

Here the problem is for a health care provider, which means we should not neglect any case of error as it might even cost the life of a person.

The first step of solving a data science problem is to observe the data and get any possible insights from it. So, let us start with understanding the data.

Understanding the Data:

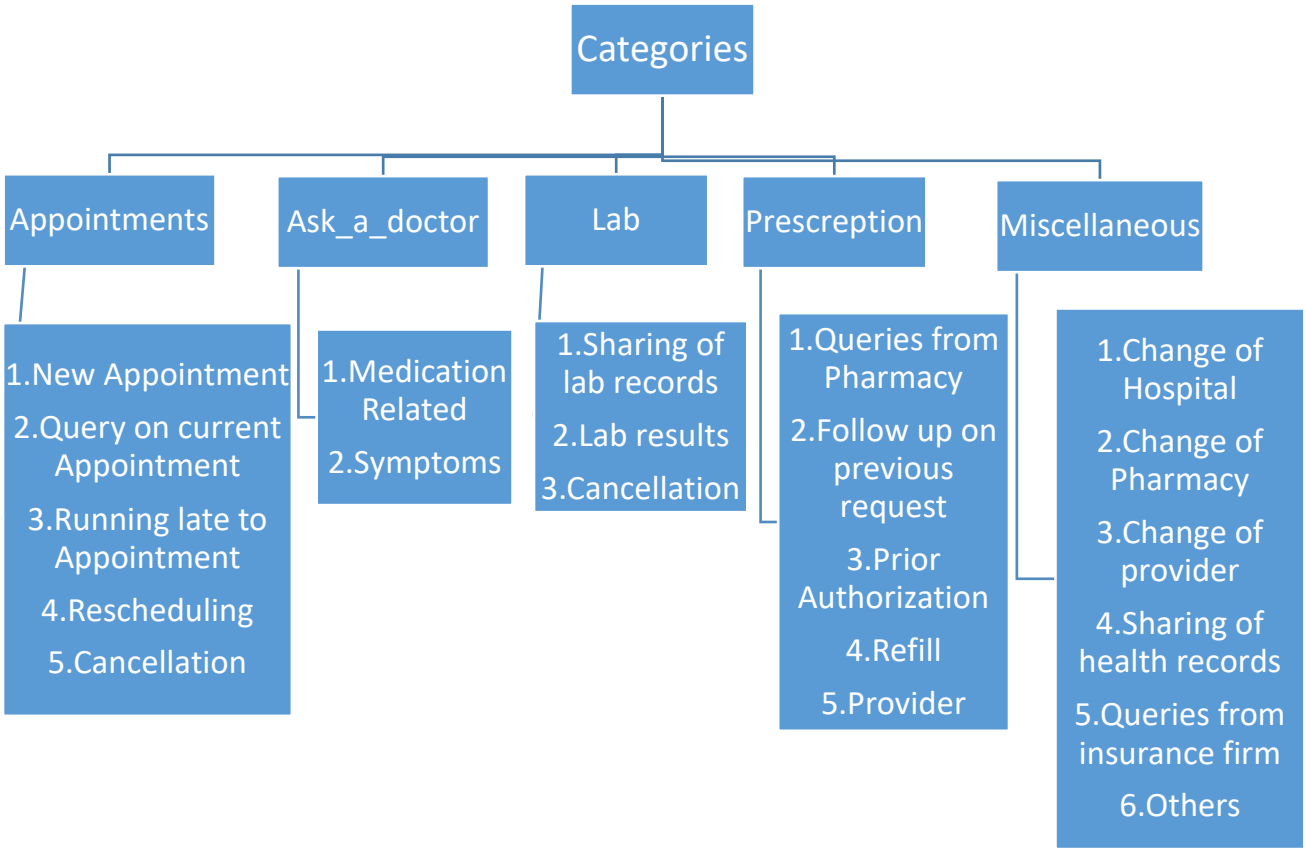
The data consists of 57280 observations and 7 variables. Let us take a look at the structure of the data.

```
> str(data)        #observe structure of data

'data.frame':   57280 obs. of 7 variables:
 $ fileid              : num 2.02e+12 2.02e+12 2.02e+12 2.02e+12 2.02e+12 ...
 $ SUMMARY             : Factor w/ 41222 levels "'Transitions' needs clarification of
                          Hospice Orders",..: 24262 16555 24435 11136 14126 4809 ...
 $ DATA                : Factor w/ 54770 levels "{\\rtf1\\ansi\\ftnbj{\\fonttbl{\\f0 \
                         \fmodern Courier New;}{\\f1 \\fswiss Arial;}}{\\colortbl ;\\re
                          d255\\green2"| __truncated__,..: 8429 8504 46400 34289 6058 .
 $ categories          : Factor w/ 8 levels "APPOINTMENTS",..: 8 3 3 7 7 1 3 7 1 7 ...
 $ sub_categories      : Factor w/ 22 levels "CANCELLATION",..: 17 9 9 11 20 16 22 20
 $ previous_appointment: Factor w/ 5 levels "No","NO","yes",..: 1 1 1 1 1 1 1 1 1 1 ..
 $ ID                  : Factor w/ 57280 levels "2015_1_1_1001",..: 38127 38128 ...
```

After observing head and structure of the data we can infer that the data is rich in text format and is unstructured. We need to clean the data before further analysing the data, i.e. we need to extract meaningful words from the “summary” and “data” variables. This is done by “Text Mining”.

Now let us observe the target variables. There are 5 distinct categories and 20 subcategories



By observing the categories and subcategories we infer that though there are 20 distinct subcategories, there are only a maximum of 5 to 6 subcategories for each category. So, instead of predicting the subcategory directly we need to predict the category first and then build separate models to predict subcategory for each category and then join them by a series of if-else conditions.

It's obvious that the variables "Field id" and "id" doesn't contribute in classifying the categories and subcategories. So, we can directly remove them.

Now in order to understand more about the data and know which words contribute more in classifying the target variables we have to pre-process the data and perform text mining.

Data Pre-Processing and Text Mining:

- First step of pre-processing is to convert the data types of each variable to required data types.
- Then we need to check if there are any missing values in the data. Here there are around 3400 missing values which is much smaller compared to no.of observations. So, we remove the observations with missing values.
- Now in order to clean the "summary" and "data" variables, a corpus is created and a series of cleaning techniques are applied on it and then it is tokenized to get meaningful words from each sentence.
- I have created a function (DF) which takes a character vector as input, performs all the cleaning operations and converts it into a data frame.
- Using "tm" package available in R, following operations are performed on the corpus by the user defined function. Let us take an example sentence and see how it is being converted after each pre-processing step.
  Eg. WCB 5/4 re: discuss injection appt

Case Folding:

In this step all the characters in a sentence are converted into lower case as they don't signify any meaning.

Eg. wcb 5/4 re: discuss injection appt

Remove Punctuation marks:

In this step punctuation marks are removed because when the sentences are tokenized punctuation marks doesn't play any role.

Eg. wcb 5 4 re discuss injection appt

Remove Numbers:

In this step numbers are removed as they can't be tokenized.

Eg. wcb re discuss injection appt

Stemming:

Documents contain different inflectional forms like tense forms and derivational forms, we perform stemming to reduce all those words to their root word.

Eg. wcb re discuss injection appt

Remove Stop Words:

Stop words are words which doesn't contribute in predicting the target variable. Every language has a standard set of stop words. Here, along with the standard stop words of English we also give some additional stop words as the "data" variable has much noise and some words that doesn't contribute in predicting the target variable.

Eg. wcb re discuss injection appt

Strip white spaces:

By removing the numbers, punctuation marks, etc... they are not removed but are replaced by empty spaces. So, now the empty spaces are to be removed, which is done in this step.

Eg. wcb re discuss injection appt

By performing all the above pre-processing steps most of the noise in the documents is removed and now the documents can be tokenized. This is done by converting the corpus into a term document matrix.

Creating a document term matrix:

Using "tm" package in R we can create a document term matrix directly from a corpus. If not mentioned weightage is taken for term frequency but in our data there are many words which are repeated mostly in many documents (especially in "data" variable) and are of less contribution to predict target variable. So, I have taken weightage as TF-IDF score which is optimal in this situation. Let us have a look at the document term matrix created from "data" variable:

```
> inspect(tdm[155:165,110:120])

<<DocumentTermMatrix (documents: 11, terms: 11)>>
Non-/sparse entries: 2/119
Sparsity           : 98%
Maximal term length: 10
Weighting          : term frequency - inverse document frequency (normalized) (tf-idf)
Sample             :

    Terms
Docs  alan      aware steep       stop sweating sweats taste unrelieved week weird
  155    0 0.00000000     0 0.00000000        0      0     0           0    0     0
  156    0 0.00000000     0 0.00000000        0      0     0           0    0     0
  157    0 0.00000000     0 0.00000000        0      0     0           0    0     0
  158    0 0.00000000     0 0.00000000        0      0     0           0    0     0
  159    0 0.00000000     0 0.00000000        0      0     0           0    0     0
  160    0 0.00000000     0 0.00000000        0      0     0           0    0     0
  161    0 0.00000000     0 0.00000000        0      0     0           0    0     0
  162    0 0.00000000     0 0.00000000        0      0     0           0    0     0
  163    0 0.00000000     0 0.00000000        0      0     0           0    0     0
  165    0 0.04470472     0 0.04882622        0      0     0           0    0     0
```

Once we clean the data we write it back into the data frame “data_cleaned” and write it as a .csv file in working directory which is taken as input file for python code.

This is how the data looks once cleaned and ready for building models

summary	data	previous_	categories	sub_categories
pt aware need rov r	phone note call patie	1	PRESCRIPTION	REFILL
mom want know fo	phone note call patie	1	ASK_A_DOCTOR	MEDICATION RELATED
pt call discuss nortr	phone note call patie	1	ASK_A_DOCTOR	MEDICATION RELATED
fyi nortryptline me	phone note call patie	1	MISCELLANEOUS	OTHERS
letter patient estab	phone note call patie	1	MISCELLANEOUS	SHARING OF HEALTH RECORDS
appt question	phone note call patie	1	APPOINTMENTS	QUERY ON CURRENT APPOINTM
dizziness double vi	phone note call patie	1	ASK_A_DOCTOR	SYMPTOMS
please refax neuro	phone note call patie	1	MISCELLANEOUS	SHARING OF HEALTH RECORDS
pt want reschedul e	phone note call patie	1	APPOINTMENTS	RESCHEDULING
phone note	mliform mliform mli	1	MISCELLANEOUS	OTHERS
rov	phone note call patie	1	APPOINTMENTS	NEW APPOINTMENT
please go ahead or	phone note call patie	1	PRESCRIPTION	PROVIDER
pt clld check work a	phone note call patie	1	APPOINTMENTS	NEW APPOINTMENT
appt sched amdb	converted care alert	1	APPOINTMENTS	NEW APPOINTMENT

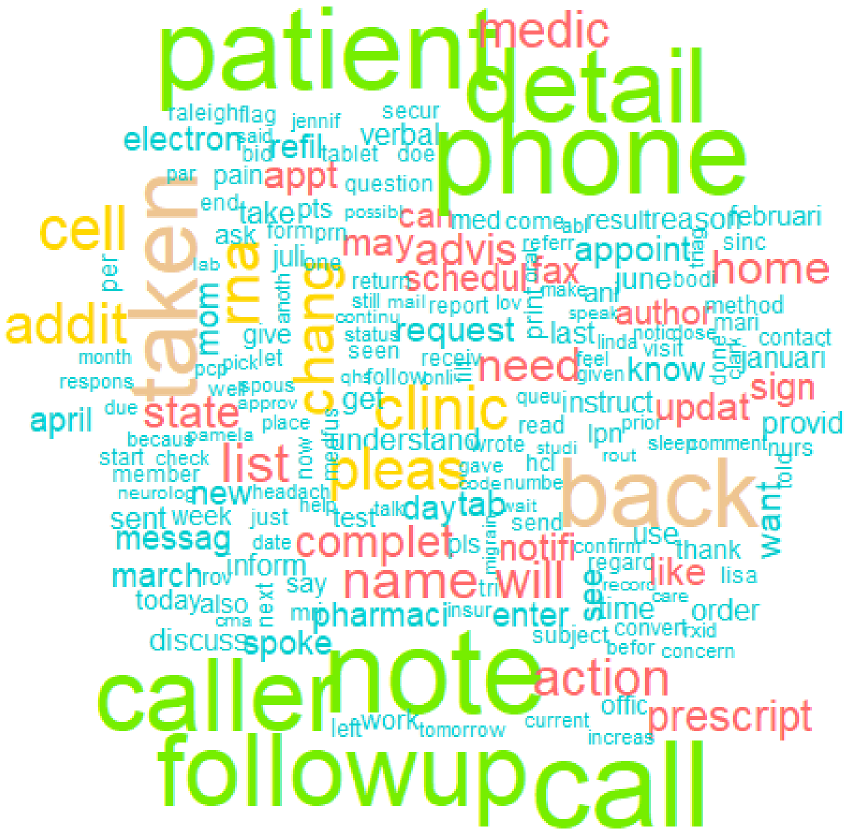
Now to get more insight on our data we have to visualize it before building our models.

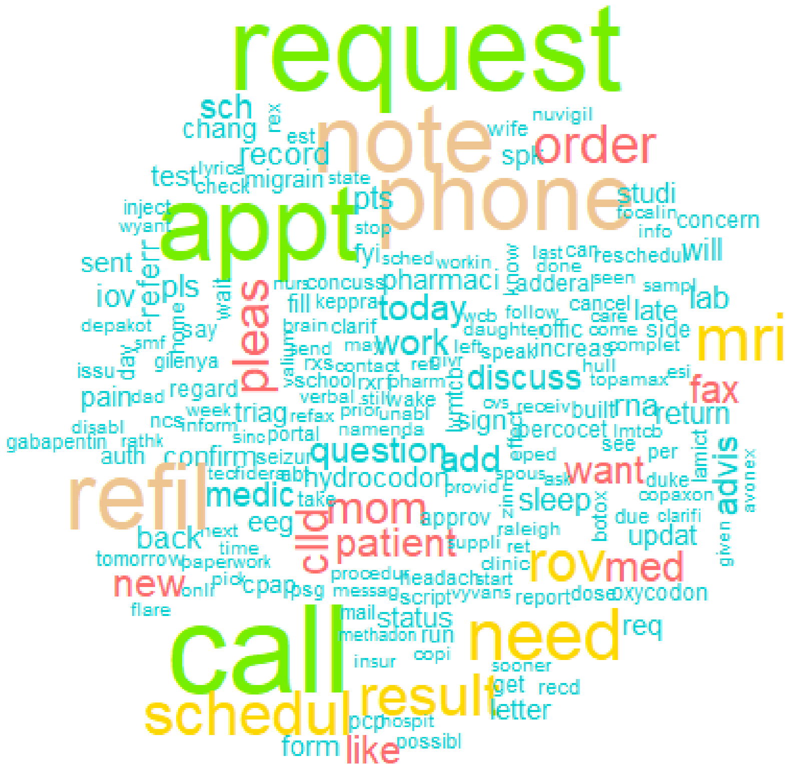
All the above pre-processing steps are wrapped in a user defined function so that they are used to prepare term document matrices which are used in making visualizations.

Data Visualizations:

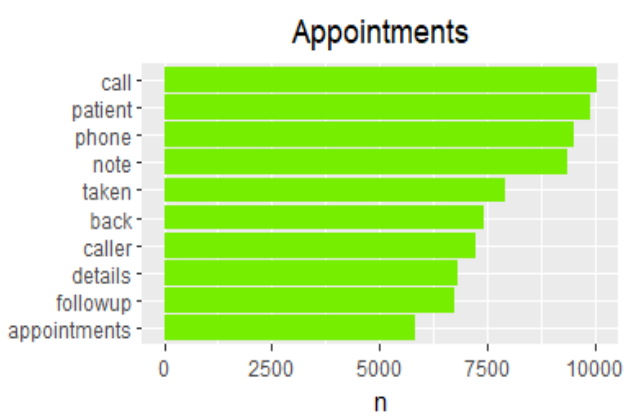
- Visualizations are the best way to get insights on dependency of target variable on independent variables.
- In our data set the independent variables are terms in the document term matrix we have created after cleaning the corpus. So, I have created 2 word clouds, one from most repeated words of “summary” variable and the other one from most repeated words of “data” variable.
- I have also created 5 bar charts representing top 10 most repeated words from each category.
- From the word clouds we can get idea on most repeated words and correct them if there are any errors in them as they effect much in the prediction of target variable.

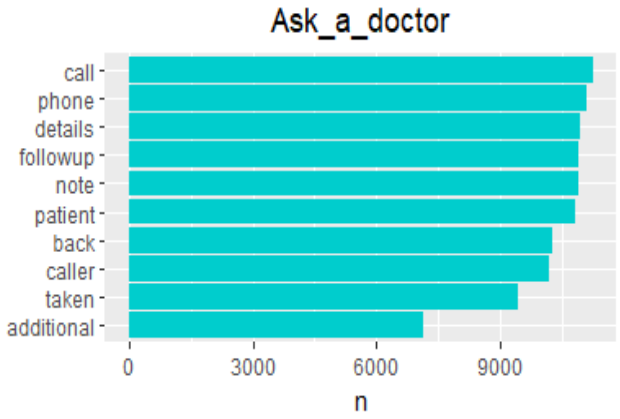
Word Clouds:

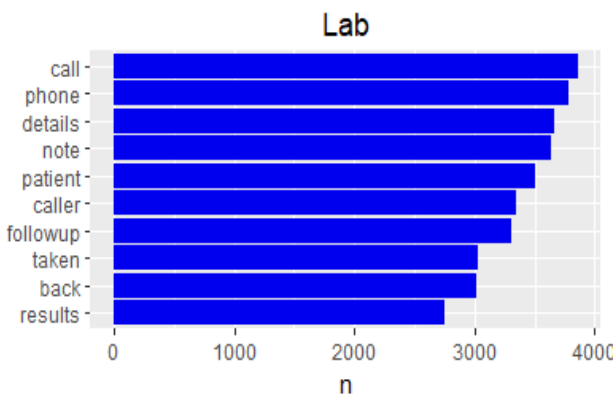

patient
detail
phone
medic
taken
cell
addit
rna
chang
clinic
back
pleas
list
state
complet
name
will
note
caller
followup
call
action
prescript
home
author
appoint
schedul
advis
fax
request
need
updat
sign
provid
know
januari
februari
pharmaci
enter
spoke
messag
april
march
today
order
time
like
notifi
want
day
tab
understand
instruct
appt
refil
electron
verbal
question
result
reason

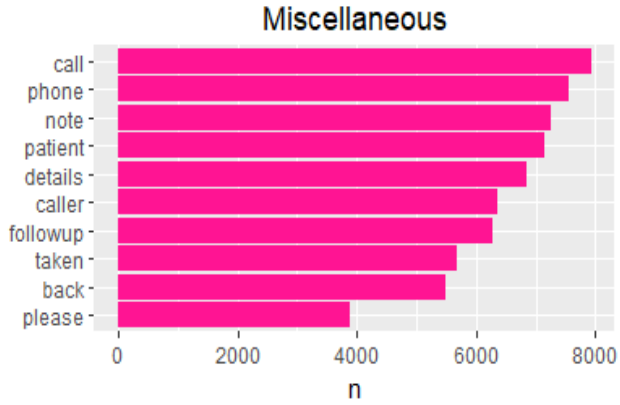

request
note
order
phone
appt
record
pleas
pharmaci
today
work
discuss
mri
fax
rna
return
advis
sleep
want
refil
medic
mom
patient
rov
med
back
new
call
need
schedul
result
like
question
hydrocodon
confirm
status
letter
form
oxycodon
percocet
adderal
concern
studi

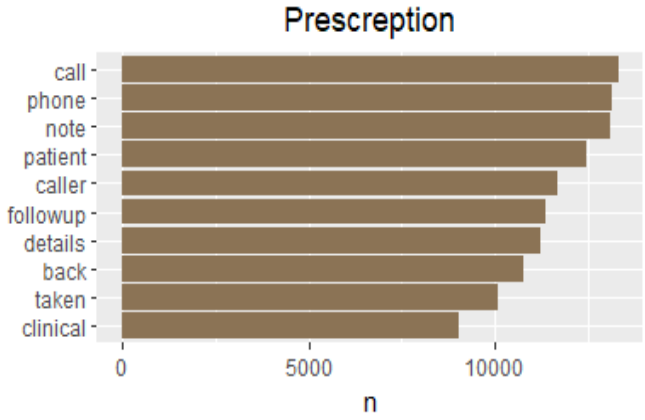
Bar Plots:











- From word clouds we infer that there are many common words among most repeated words in “summary” and “data” variables. So, we need to take care of them while merging both the corpuses.
- From bar plots we can say that there are many common words among most repeated words in all the categories. So, we have to take TF-IDF as our weightage scheme while preparing the document term matrix.

As we have completed cleaning our data and got some insights from our data we can move on to preparing our models for predicting categories and sub_categories.

Model building starts with preparing train and test data sets, then training the machine learning model with train set and checking its performance with test data set.

Model Building:

Building a machine learning model is faster and easier in python compared to R as we can build our models using a sparse matrix in python and hence there is no need of decreasing the sparsity which results in more accurate model.

So, I have done all cleaning part in R and my R code when run writes a cleaned .csv file into the working directory which is taken as input for my python code in which I have built the machine learning models.

Preparing train and test data sets:

First step of model building is dividing the data into train and test sets. Using "sklearn" in python we can create a corpus of both "summary" and "data" together.

Our first train and test data sets are to predict the "category" variable. We select directly a random sample of 70% of data as train and remaining 30% as test and convert them into sparse matrices using "sklearn" library.

Similarly we prepare train and test data sets for each category to predict "sub_category" variable.

Once we have prepared our train and test data sets we can move on to building our classification models. I have chosen "Naïve Bayes Classifier" and "SVM Classifier" as machine learning models.

Naïve Bayes Classifier:

The primary assumption of Naïve Bayes classifier is that all the variables are equally important and independent of each other. Our data is text data and obviously every term is independent of any other term. So, Naïve Bayes classifier is better choice compared to other machine learning algorithms.

Upon training the data with Naïve Bayes classifier of "sklearn" to predict categories we are getting an accuracy of "73%" when tested on test set. When we try to predict the subcategory directly from entire data set we are getting a poor accuracy of "53%".

Now when we build separate models for each category using Naïve Bayes classifier we are getting an accuracy of (73, 77, 91, 78, 73)% for ("prescription", "appointments", "ask a doctor", "lab", "miscellaneous") respectively. This gives an average accuracy of "78%" in predicting subcategories.

Thus this justifies why we need to build separate models to predict subcategories once we have predicted our category variable.

SVM Classifier:

Text data is inherently discrete, sparse and high dimensional and as SVM performs well with that kind of data, "SVM classifier" had been used historically on text classification models and that makes it a better choice for our data set.

Similar to Naïve Bayes classifier we train the SVM classifier with the same train set that we used for training Naïve Bayes classifier and when tested on test data we are getting an accuracy of “81%” for category variable and “73%” for subcategory variable when predicted using all categories.

When we build separate models for each category we are getting an accuracy of (92,88,90,90,84)% for (“prescription”, “appointments”, “ask a doctor”, “lab”, “miscellaneous”) respectively which gives us an average accuracy of “89%” which is pretty high compared to Naïve Bayes classifier or when predicted using entire data set.

Conclusion:

We finally freeze the SVM model as it is more accurate with an accuracy of 81 % for category variable and an average accuracy of 89% for subcategory variable.