JANUARY 20, 2018

# HEALTHCARE SERVICES

PRESENTED BY: YASH MITTAL

PRESENTED TO: EDWISOR.COM

# CONTENTS

## 1. Problem Statement:

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, general doctor advice 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).

**It is a text classification problem.**

## 2. Data Used:

We are provided with a csv file which contain 57280 rows and 7 columns. These 7 columns are:

- fileid
- summary
- data
- categories
- sub_categories
- previous_appointment
- id

categories and sub_categories are our target variables which we have to predict via other variables. fileid and id are used for numbering of the data in csv, and these are of no use in predicting our target variables. We will remove these 2 variables and then we will be working with 5 variables. We are now only left with 3 variables that are summary, data and previous_appointment which can be used for prediction.

### 3. Exploration of Numerical variables:

We have no numerical variables on which we need to do analysis as the numerical variables fileid and id are already left behind.

## 4. Exploration of Categorical variables:

We have 3 categorical variables in data:

- previous_appointment
- category
- sub_category

In previous_appointment we get counts of ‘no’ and ‘yes’ as:

```
no     57083
yes      195
Name: previous_appointment, dtype: int64
```

There are 2 blank entries in previous_appointment variable.

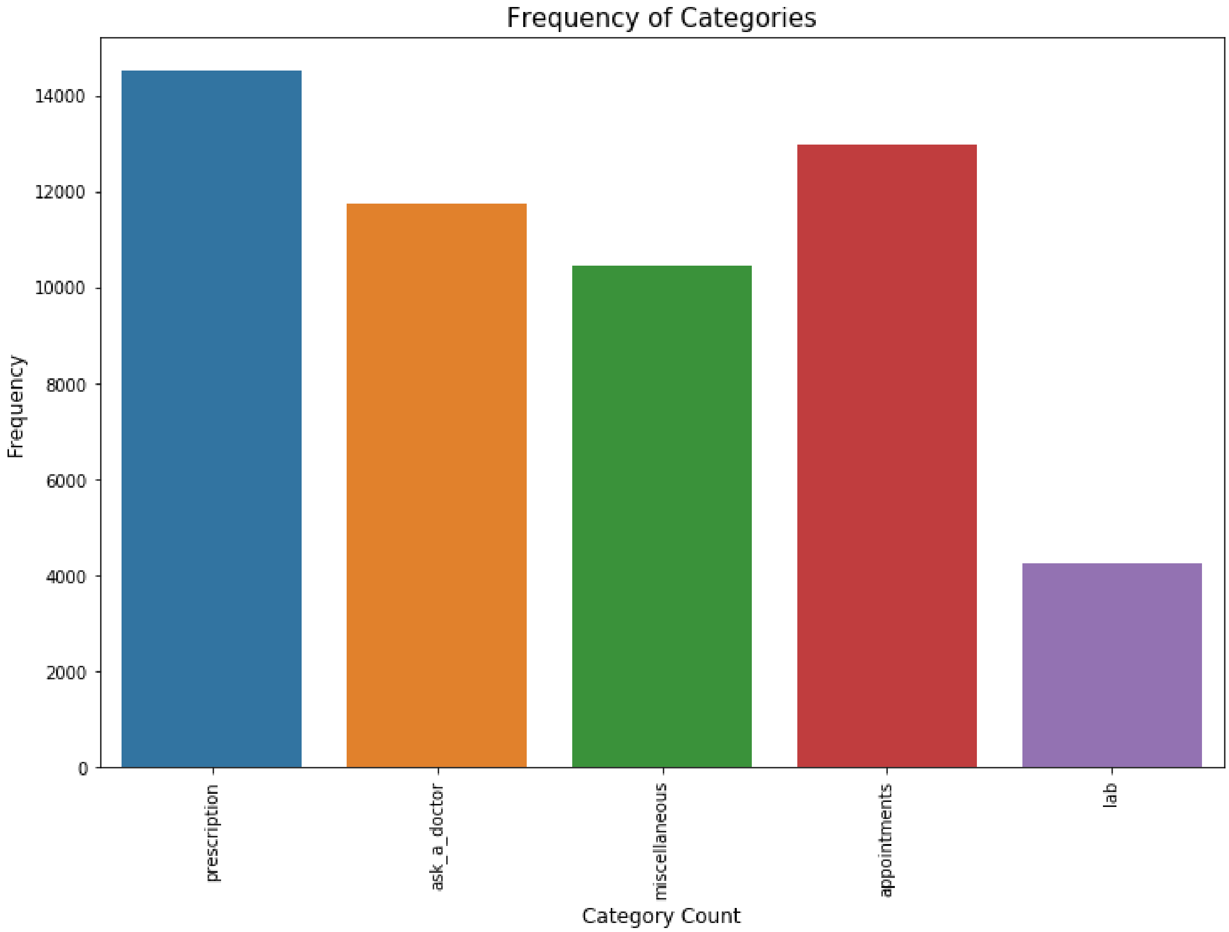
After removal of all incomplete rows we get value counts for category and sub category as:

```
prescription     14500
appointments     12960
ask_a_doctor     11744
miscellaneous    10462
lab               4246
Name: categories, dtype: int64
```

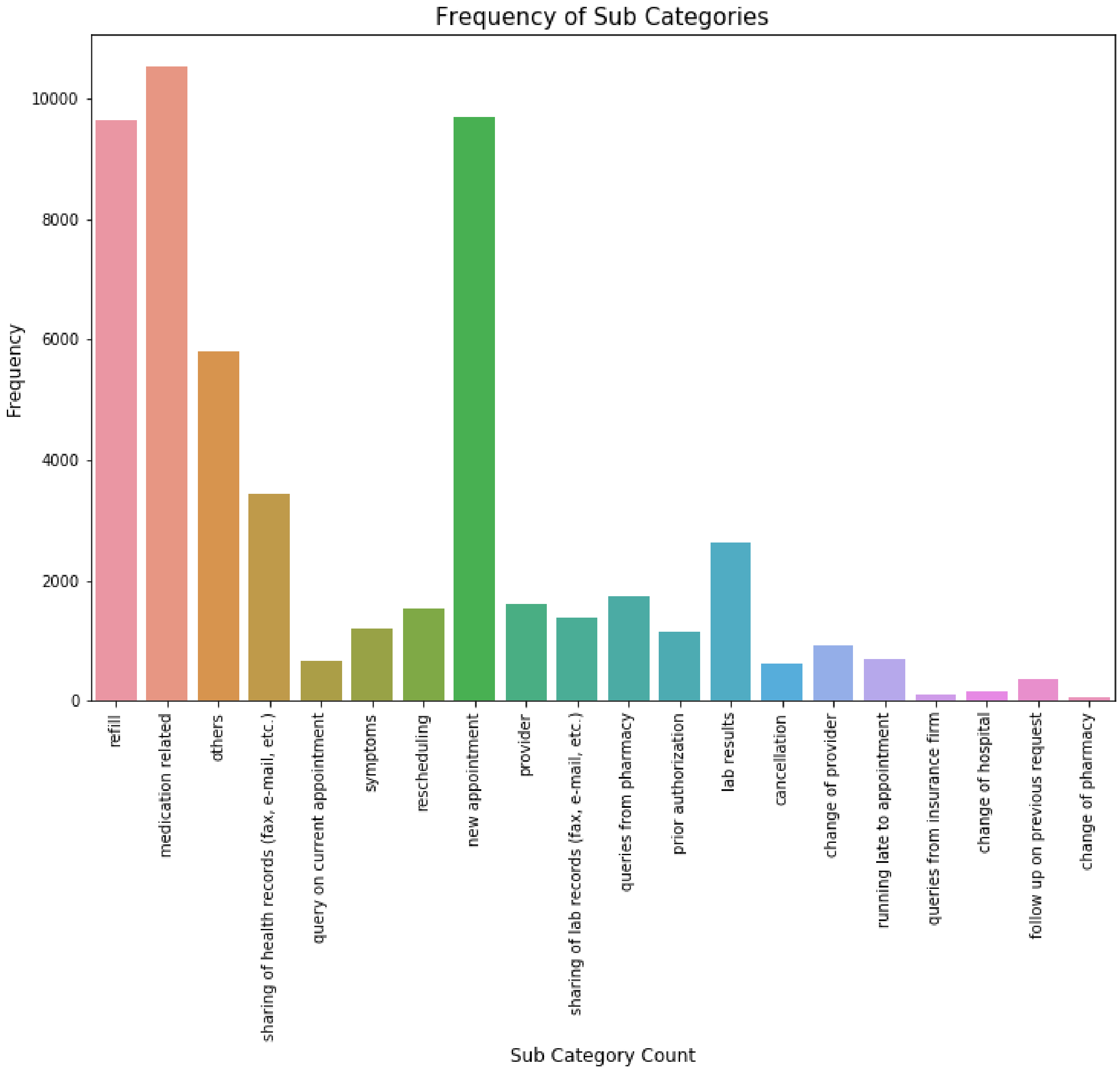
```
medication related                               10547
new appointment                                   9713
refill                                            9665
others                                            5796
sharing of health records (fax, e-mail, etc.)     3435
lab results                                       2628
queries from pharmacy                             1722
provider                                          1608
rescheduling                                      1520
sharing of lab records (fax, e-mail, etc.)        1386
symptoms                                          1197
prior authorization                               1155
change of provider                                 928
running late to appointment                        694
```

```
query on current appointment                      652
cancellation                                      613
follow up on previous request                     350
change of hospital                                142
queries from insurance firm                       107
change of pharmacy                                 54
Name: sub_categories, dtype: int64
```

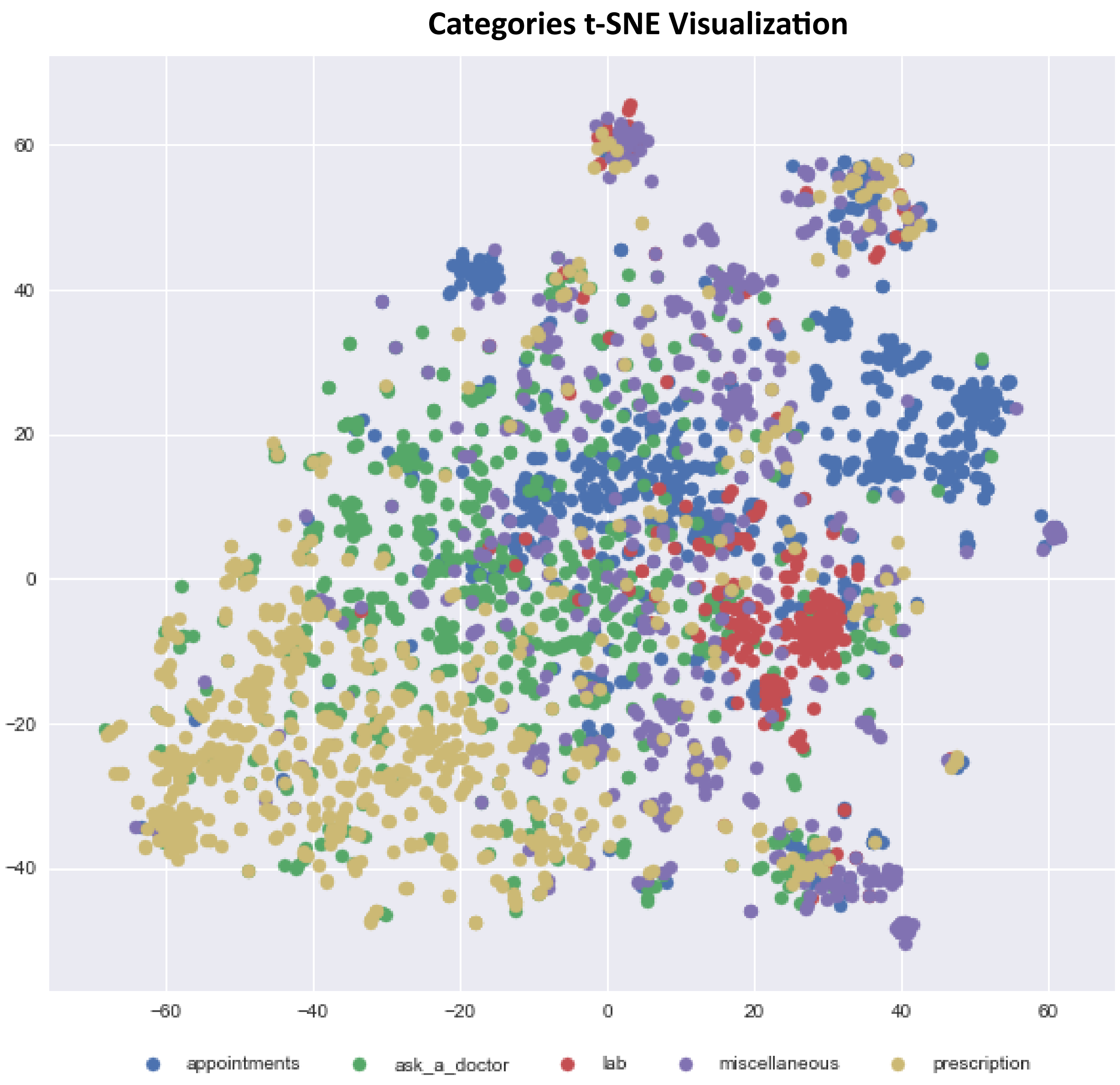
## CATEGORY DISTRIBUTION



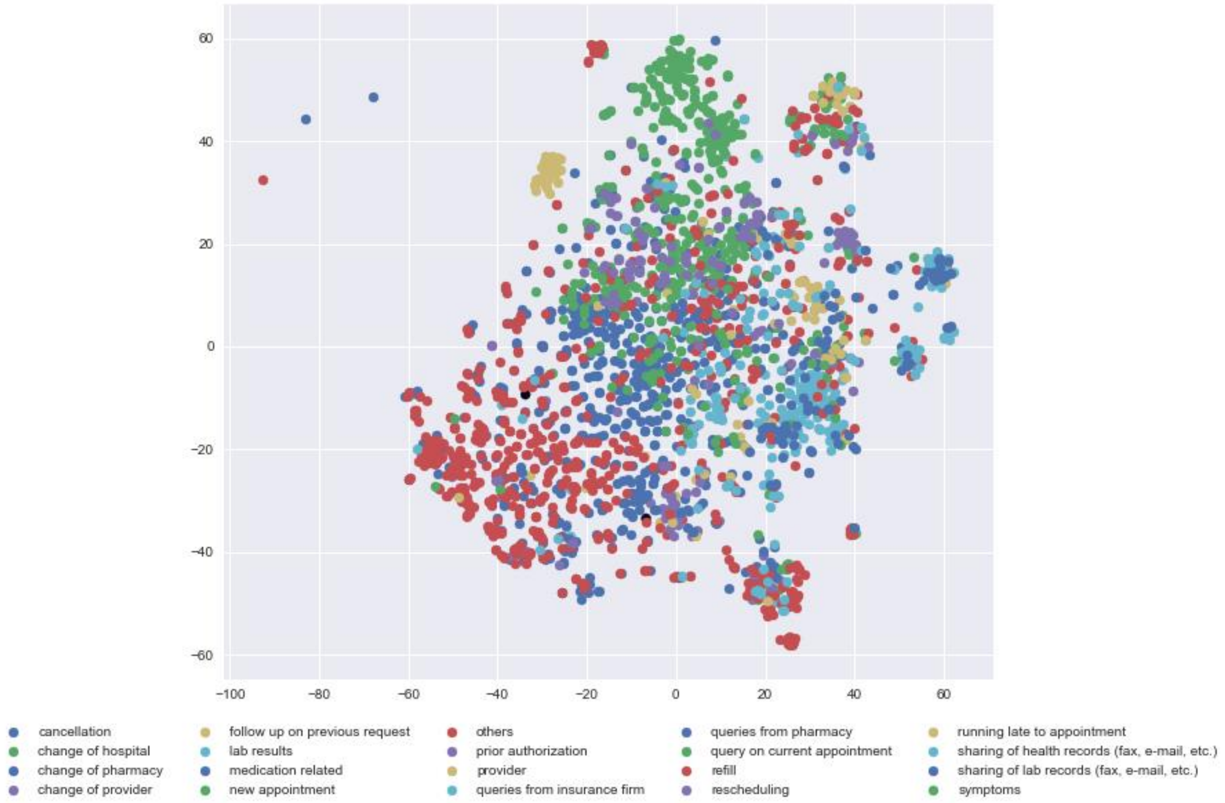
## SUB CATEGORY DISTRIBUTION



We can also visualize this high dimensional data of categories and sub categories using t-SNE.



## Sub Categories t-SNE Visualization



## 5 . Hypothesis Testing:

The P-value approach involves determining "likely" or "unlikely" by determining the probability — assuming the null hypothesis were true — of observing a more extreme test statistic in the direction of the alternative hypothesis than the one observed. If the P-value is small, say less than (or equal to) α, then it is "unlikely." And, if the P-value is large, say more than α, then it is "likely."

If the P-value is less than (or equal to) α, then the null hypothesis is rejected in favour of the alternative hypothesis. And, if the P-value is greater than α, then the null hypothesis is not rejected.

### 5.1 **Chi-Square Test:**

We will use chi-square test to find the dependency of categorical variables on each other.

- **Hypothesis testing on previous_appointment and categories**:
  We will assume that they are independent and when we perform chi square test by assuming alpha as 0.05 we get value of p as 7.1869681817705634e-16 which is quite smaller to alpha. So our hypothesis failed and they are dependent on each other
- **Hypothesis testing on categories and sub_categories**:
  We will assume that they are independent and when we perform chi square test by assuming alpha as 0.05 we get value of p as 0 which is too smaller to alpha. So our hypothesis failed and they are strongly dependent on each other.

## 6. Data Pre-processing:

DATA variable contains a lot of junk in it which needs to be cleared before putting the data into various machine learning models. Example of a cell is:

```
'{\\rtf1\\ansi\\ftnbj{\\fonttbl{\\f0 \\fswiss Arial;}}{\\colortbl ;\\red25
5\\green255\\blue255 ;\\red0\\green0\\blue255 ;\\red0\\green0\\blue0 ;\\re
d0\\green0\\blue255 ;\\red0\\green128\\blue0 ;}{\\stylesheet{\\f0\\fs20\\c
f3\\cb1 Normal;}{\\cs1\\additive\\cf3\\cb1 Default Paragraph Font;}}\\marg
l1440\\margr1440\\margt540\\margb1440\\headery540\\footery720\\formshade\\
sectd\\marglsxn1440\\margrsxn1440\\margtsxn540\\margbsxn1440\\headery540\\
footery720\\sbkpage\\pgncont\\plain\\plain\\fs20\\pard\\plain\\fs20\\cf0\\
fs24\\sscharaux1\\b Phone Note \\fs20\\b0\\par\\fs24\\b\\par CALL FROM PHA
RMACY\\par  \\fs20 Caller Name: \\b0xxxx-xxxxar\\b Caller: \\b0 right sour
ce\\par\\b\\par Reason for Call: \\b0 Details: 866-4862668\\par Requesting
 verbal bc original rx was an ink blurr.\\par\\par Plegridy\\par\\b Call T
aken by: \\b0 xxxx-xxxx ,  January  5, 2015 2:12 PM\\par\\b Follow-up Deta
ils: \\b0 2nd call concerning verbal order for Plegridy\\par\\b Follow-up
by: \\b0 Paulette Lee,  January  5, 2015 4:25 PM\\par\\b Additional Follow
-up Details: \\b0 3rd call concerning verbal order for Plegridy, pt was to
 start taking it today. \\par\\b Additional Follow-up by: \\b0 Paulette Le
e,  January  6, 2015 8:35 AM\\par\\fs24\\b\\par Clinical List Changes\\par
\\fs20\\par Medications Updated:\\par\\b0 Added new medication of PLEGRIDY
 STARTER PACK 63 & 94 MCG/0.5ML SOPN (PEGINTERFERON BETA-1A) Inject 63mcg
SQ Day 1, then Inject 94 mcg Day 15, then start maintence dose 125 mcg the
reafter q 14 days - Signed\\par Removed medication of AVONEX PEN 30 MCG/0.
5ML KIT (INTERFERON BETA-1A) inject once a week via Avonex Pen\\par Added
new medication of PLEGRIDY 125 MCG/0.5ML SOPN (PEGINTERFERON BETA-1A) Inje
ct 125 mcg SQ q 14 days - Signed\\par Rx of PLEGRIDY STARTER PACK 63 & 94
MCG/0.5ML SOPN (PEGINTERFERON BETA-1A) Inject 63mcg SQ Day 1, then Inject
94 mcg Day 15, then start maintence dose 125 mcg thereafter q 14 days;  #1
 Package x 0;  Signed;  Entered by: Lisa Jones RN;  Authorizedxxxx-xxxxWES
T '
```

This contain various '\\' signs with a lot of garbage data with them.

Following code is used to clean data from all the rows:

```
%%time
for i in range(0,df.shape[0]):
    print i
    df.iloc[i,2]=re.sub(r'\\.*? |\\.*',' ',df.iloc[i,2])
    df.iloc[i,2]=re.sub(r'xxxx|xxx|-',' ',df.iloc[i,2])
    df.iloc[i,2]=re.sub(r'.+;}',' ',df.iloc[i,2])
    df.iloc[i,2] = re.sub(r'[a-zA-Z]+\s+\d{1,2},\s+\d{4}\s+\d{1,2}:\d{2}\s+(AM|PM){0,1}',' ',df.iloc[i,2])
```

Data of dates is also removed since we don’t need them while building our bag of words model. Now the clean data is saved in processed.csv file so that next time we use data we don’t need to process that again.

There are also missing values in our data which needs to be removed.

```
df1.isnull().sum()
```

```
SUMMARY                 3347
DATA                       0
categories                 0
sub_categories             0
previous_appointment       2
dtype: int64
```

There are 3347 null values in SUMMARY variable and 2 null values in previous_appointment.

```
df1=df1.dropna()
```

All null values will be dropped from the dataframe.

We will also convert all data of categories and sub_categories into lower case so that they doesn’t get treated as different.

In some rows values of categories and sub categories are missing and that is filled with word ‘junk’. Those rows will create problem while prediction so we will also drop those rows. After removing all unwanted rows we are now left with 53912 rows.

Now, we will make use of 2 columns DATA and SUMMARY and will make a single column total_data by merging them.

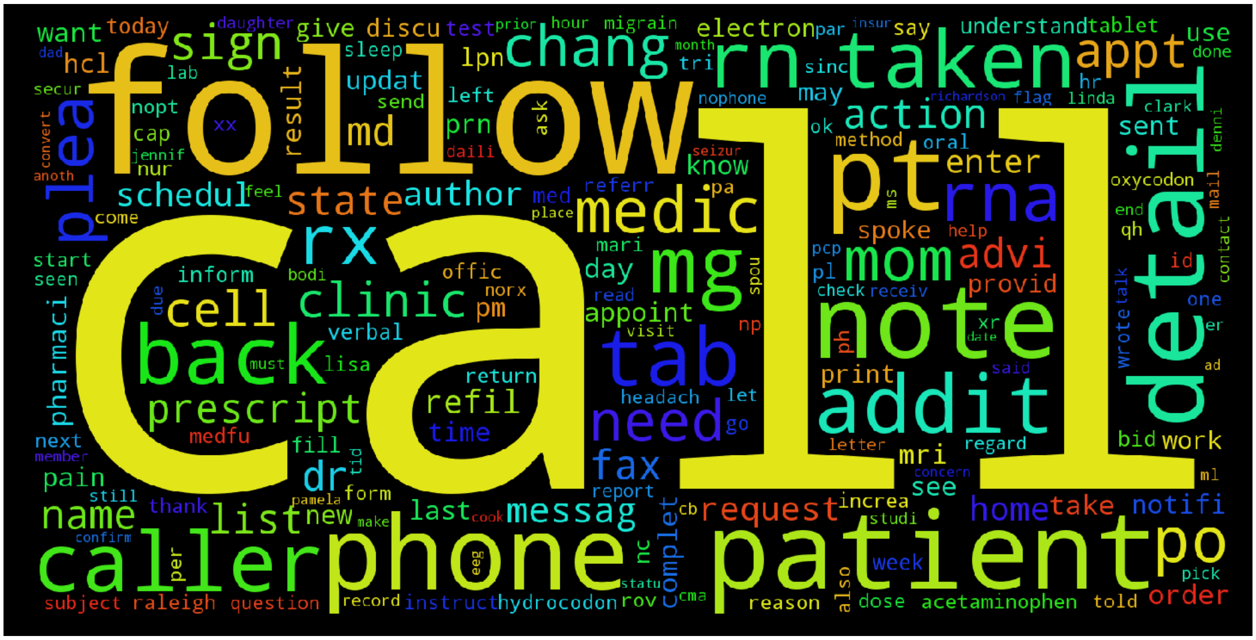
This total data will be cleaned now and values of previous_appointment variable will also be added into it. Data like punctuation marks are removed and only plain text is kept. Data is converted into lower case. All the extra whitespaces are removed.

We will also perform stemming there so that words like ‘patient’ and ‘patients’ are treated as same. The following code is doing that and making corpus for our upcoming model:

```
%%time
corpus=[]
fdf['clean_data']=""
for i in range(0, fdf.shape[0]):
    print(i)
    review = re.sub('[^a-zA-Z]', ' ', fdf['total_data'][i])
    review = review.lower()
    review = review.split()
    ps = PorterStemmer()
    review = [ps.stem(word) for word in review if not word in set(stopwords.words('english'))]
    review = ' '.join(review)
    review=review+' '+fdf['previous_appointment'][i].lower()
    fdf['clean_data'][i]=review
```

This data is saved to file final.csv so that we don't need to do same processing again when coming back on work.

Corpus visualization using word cloud:



We can see some most commonly words in word cloud like call, patient, caller phone, follow, detail, etc.

## 7. Building Predictive Models:

### 7.1. SGD Classifier:

This estimator implements regularized linear models with stochastic gradient descent (SGD) learning: the gradient of the loss is estimated each sample at a time and the model is updated along the way with a decreasing strength schedule (aka learning rate). For best results using the default learning rate schedule, the data should have zero mean and unit variance.

This implementation works with data represented as dense or sparse arrays of floating point values for the features. The model it fits can be controlled with the loss parameter; by default, it fits a linear support vector machine (SVM).

I am creating pipeline for this using Countvectorizer and Tf-idf:

```
category_clf = Pipeline([('vect', CountVectorizer(ngram_range=(1,2))), ('tfidf', TfidfTransformer()),
('clf', SGDClassifier(loss = 'modified_huber', alpha=0.0001, penalty = 'l2', n_iter = 100, random_state = 0))])
```

The data is then divided into training and testing data in proportion of 80:20 using stratified sampling.

Classifier is trained and saved on harddisk:

```
with open('SGD_category.pkl', 'wb') as f:
    pickle.dump(category_clf, f)
```

We obtain accuracy of 82.5% through this classifier. In cross validation check we found that accuracy is around 81.7% which is quiet close to it.

Following is the metrices classification report:

```
              precision    recall  f1-score   support

 appointments       0.88      0.86      0.87      2693
 ask_a_doctor       0.76      0.78      0.77      2305
          lab       0.85      0.76      0.80       820
miscellaneous       0.76      0.79      0.77      2099
 prescription       0.85      0.87      0.86      2866

  avg / total       0.82      0.82      0.82     10783
```

Now for predicting sub categories, categories will also play an important role so we will add categories column also to our corpus for predicting sub categories and save merged column clean_data_categories with dataframe to final_sub_categories.csv.

For predicting sub categories again we split data into training and test data in proportion of 80:20. Classifier of sub category prediction is made:

```
sub_category_clf = Pipeline([('vect', CountVectorizer(ngram_range=(1,2))), ('tfidf', TfidfTransformer()),
('clf', SGDClassifier(loss = 'modified_huber', alpha=1e-05 , penalty = 'l2', n_iter = 100, random_state = 0))])
```

We train the model and save it with:

```
with open('SGD_sub_category.pkl', 'wb') as f:
  pickle.dump(sub_category_clf, f)
```

For predicting sub category I predict category first and then added that category to the corpus to predict sub category. Through this I achieved accuracy of 80.8% almost 81% on sub category prediction.

Metrices classification report for sub category prediction is:

```
                                               precision    recall  f1-score   support

                                  cancellation       0.87      0.58      0.69       123
                            change of hospital       0.00      0.00      0.00        28
                            change of pharmacy       0.00      0.00      0.00        11
                            change of provider       0.59      0.59      0.59       186
                 follow up on previous request       0.55      0.17      0.26        70
                                   lab results       0.84      0.90      0.87       526
                            medication related       0.80      0.86      0.83      2110
                               new appointment       0.83      0.90      0.86      1943
                                        others       0.75      0.79      0.77      1159
                           prior authorization       0.78      0.77      0.77       231
                                      provider       0.69      0.68      0.68       322
                   queries from insurance firm       1.00      0.33      0.50        21
                         queries from pharmacy       0.97      0.96      0.96       344
                  query on current appointment       0.52      0.20      0.29       130
                                        refill       0.89      0.93      0.91      1933
                                  rescheduling       0.67      0.59      0.62       304
                   running late to appointment       0.99      0.97      0.98       139
sharing of health records (fax, e-mail, etc.)       0.75      0.76      0.75       687
   sharing of lab records (fax, e-mail, etc.)       0.76      0.60      0.67       277
                                      symptoms       0.45      0.12      0.19       239

                                   avg / total       0.80      0.81      0.80     10783
```

### 7.2. Multinomial Naïve Bayes:

The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work.

I am creating pipeline for this using Countvectorizer and Tf-idf:

```
cat_text_NB = Pipeline([('vect', CountVectorizer(ngram_range=(1,2))),('tfidf', TfidfTransformer()),('clf',
MultinomialNB(alpha=1e-2)),])
```

The data is then divided into training and testing data in proportion of 80:20 using stratified sampling.

Classifier is trained and saved on harddisk:

```
with open('category_NB.pkl', 'wb') as f:
    pickle.dump(cat_text_NB, f)
```

Here we achieve accuracy of 77.3%

Now for predicting sub categories again we split data into training and test data in proportion of 80:20. Classifier of sub category prediction is made:

```
sub_cat_NB = Pipeline([('vect', CountVectorizer()),('tfidf', TfidfTransformer()),('clf', MultinomialNB()),])
```

Model is trained and saved as:

```
with open('sub_category_NB.pkl', 'wb') as f:
    pickle.dump(sub_cat_NB, f)
```

For predicting sub categories we predict categories first of our test data and use that also to predict our sub categories. The accuracy achieved here is 54.9% almost 55% which is quiet low.

## 8. Conclusion:

In the 2 models we prepared, our 1st model is quiet good as compared to our 2nd model so SGD classifier is very good in predicting categories and sub categories with respect to Multinomial Naïve Bayes. SGD classifier is giving around 82% accuracy in prediction of categories and 81% accuracy in prediction of sub categories.

## 9. Recommendations:

We have seen that our SGD classifier failed to predict sub categories like change of pharmacy and change of hospital. We need more data for these labels or we can use techniques like SMOTE (Synthetic Minority Over-Sampling Technique) for getting a more balanced dataset. We can study values of DATA and SUMMARY columns more carefully to remove words which are repeated almost in each row like call. Better the data better will be the prediction.