ABOUT NOTEBOOK

The objective of this Notebook is to predict the sentiment of the drug Users, according to their reviews and various other features like the condition they are suffering from, the rating of the drug used, Date of the usage, and others.

link to website:-
https://www.kaggle.com/code/harshjain123/drugs-review-sentiment/notebook#notebook-container

Steps Performed

- DESCRIPTIVE STATISTICS
- DATA VISUALIZATION
- DATA PREPROCESSING & FEATURE ENGINEERING
- LIGHT GBM MODEL BUILDING

About Dataset

- The Drug Review Dataset is taken from the UCI Machine Learning Repository. This Dataset provides patient reviews on specific drugs along with related conditions and a 10-star patient rating reflecting the overall patient satisfaction. The data was obtained by crawling online pharmaceutical review sites. The Drug Review Data Set is of shape (161297, 7) i.e. It has 7 features including the review and 161297 Data Points or entries.
- The features are 'drugName' which is the name of the drug, 'condition' which is the condition the patient is suffering from, 'review' is the patients review, 'rating' is the 10-star patient

rating for the drug, 'date' is the date of the entry and the 'usefulcount' is the number of users who found the review useful.

- Here the sentiment of the review is the target variable that needs to be predicted. here we can notice that the sentiment of any review is not given, so we have to give the sentiment to the rating first and then use it as the target variable.

Let's Start the Implementation

Import Required Libraries and Load Dataset

In [1]:

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
df =
pd.read_csv('../input/kuc-hackathon-winter-2018/drugsComTrain_raw.csv')
test =
pd.read_csv('../input/kuc-hackathon-winter-2018/drugsComTest_raw.csv')
df.head()
```

Out[2]:

	uniqueID	drugName	condition	review	rating	date	usefulCount
0	206461	Valsartan	Left Ventricular Dysfunction	"It has no side effect, I take it in combinati...	9	20-May-12	27
1	95260	Guanfacine	ADHD	"My son is halfway through his fourth week of ...	8	27-Apr-10	192
2	92703	Lybrel	Birth Control	"I used to take another oral contraceptive, wh...	5	14-Dec-09	17
3	138000	Ortho Evra	Birth Control	"This is my first time using any form of birth...	8	3-Nov-15	10
4	35696	Buprenorphine / naloxone	Opiate Dependence	"Suboxone has completely turned my life around...	9	27-Nov-16	37

```
# as both the dataset contains same columns we can combine them for better analysis

data = pd.concat([df, test])
data.head()
```

Out[3]:

	uniqueID	drugName	condition	review	rating	date	usefulCount
0	206461	Valsartan	Left Ventricular Dysfunction	"It has no side effect, I take it in combinati...	9	20-May-12	27
1	95260	Guanfacine	ADHD	"My son is halfway through his fourth week of ...	8	27-Apr-10	192
2	92703	Lybrel	Birth Control	"I used to take another oral contraceptive, wh...	5	14-Dec-09	17
3	138000	Ortho Evra	Birth Control	"This is my first time using any form of birth...	8	3-Nov-15	10
4	35696	Buprenorphine / naloxone	Opiate Dependence	"Suboxone has completely turned my life around...	9	27-Nov-16	37

Descriptive Statistics

In [4]:

```
# describing the data

data.describe()
```

Out[4]:

	uniqueID	rating	usefulCount
count	215063.000000	215063.000000	215063.000000
mean	116039.364814	6.990008	28.001004
std	67007.913366	3.275554	36.346069
min	0.000000	1.000000	0.000000
25%	58115.500000	5.000000	6.000000
50%	115867.000000	8.000000	16.000000
75%	173963.500000	10.000000	36.000000
max	232291.000000	10.000000	1291.000000

```
# taking out information from the data

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 215063 entries, 0 to 53765
Data columns (total 7 columns):
 #   Column       Non-Null Count   Dtype 
---  ------       --------------   ----- 
 0   uniqueID     215063 non-null  int64 
 1   drugName     215063 non-null  object
 2   condition    213869 non-null  object
 3   review       215063 non-null  object
 4   rating       215063 non-null  int64 
 5   date         215063 non-null  object
 6   usefulCount  215063 non-null  int64 
dtypes: int64(3), object(4)
memory usage: 13.1+ MB
```

In [6]:

```
# get the datatype of columns

data.dtypes
```

Out[6]:

```
uniqueID        int64
drugName       object
condition      object
review         object
rating          int64
date           object
usefulCount     int64
dtype: object
```

```
# checking if the data contains any NULL values

data.isnull().any()
```

Out[7]:

```
uniqueID       False
drugName       False
condition       True
review         False
rating         False
date           False
```

```
usefulCount    False
dtype: bool
```

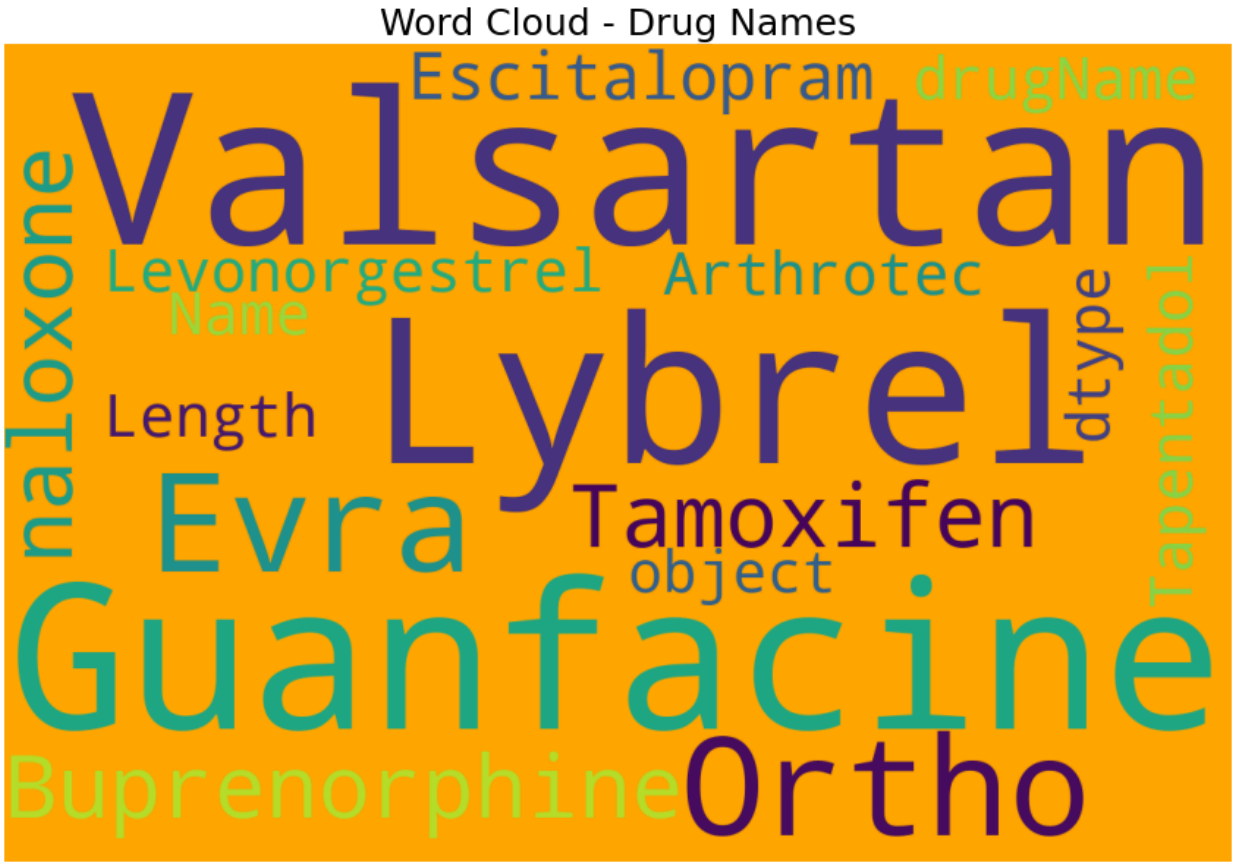
DATA VISUALIZATION

VISUALIZATION OF DRUG NAMES / RATINGS / CONDITIONS

In [8]:

```
# let's see the words cloud for the reviews

# most popular drugs

from wordcloud import WordCloud
from wordcloud import STOPWORDS

stopwords = set(STOPWORDS)

wordcloud = WordCloud(background_color = 'orange', stopwords = stopwords,
width = 1200, height = 800).generate(str(data['drugName']))

plt.rcParams['figure.figsize'] = (15, 15)
plt.title('Word Cloud - Drug Names', fontsize = 25)
print(wordcloud)
plt.axis('off')
plt.imshow(wordcloud)
plt.show()
```

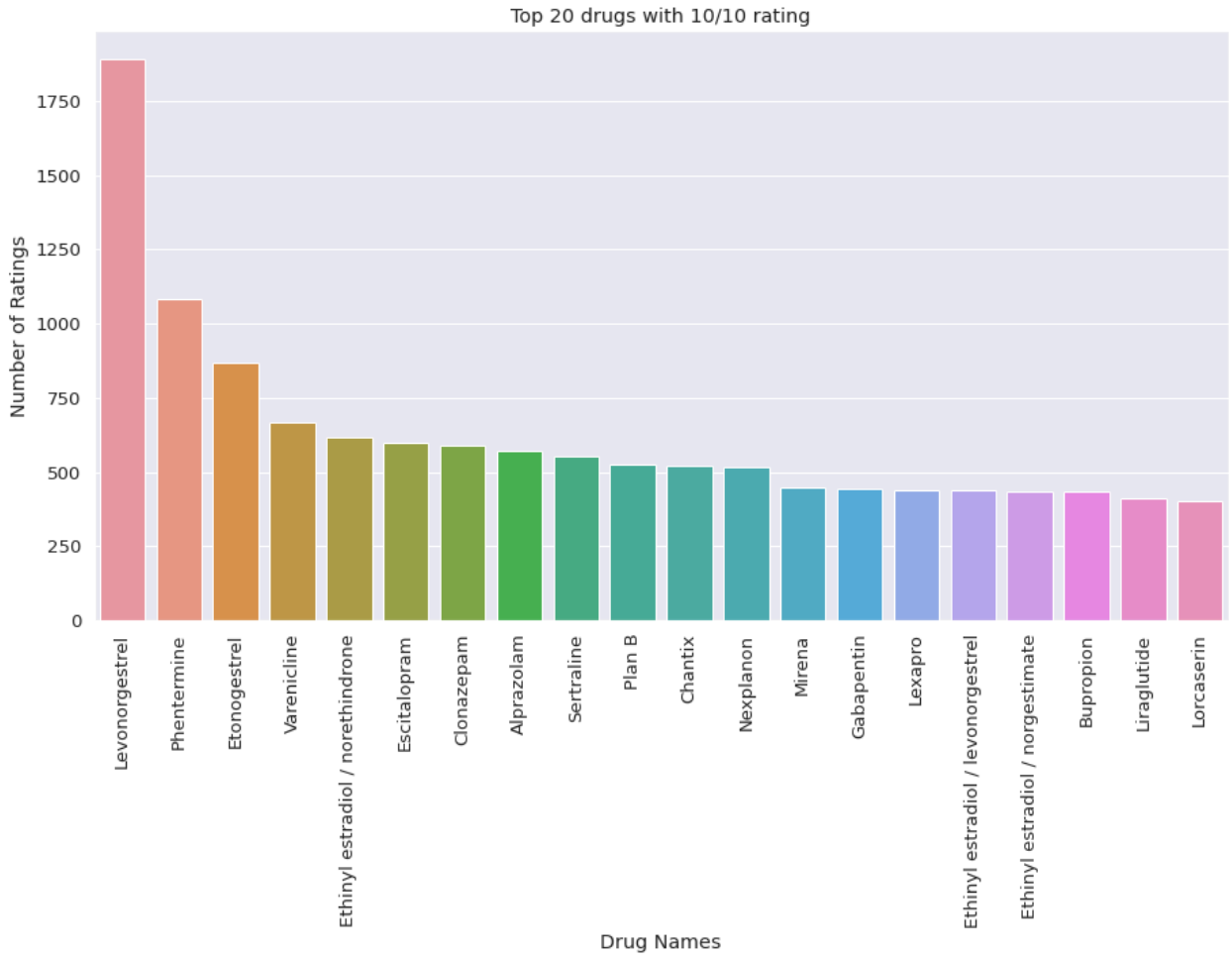
```
wordcloud.wordcloud.WordCloud object at 0x7f7cfbc81f90>
```



- This is a word cloud for the DRUG NAMES

```
# This barplot shows the top 20 drugs with the 10/10 rating

# Setting the Parameter
sns.set(font_scale = 1.2, style = 'darkgrid')
plt.rcParams['figure.figsize'] = [15, 8]

rating = dict(data.loc[data.rating == 10, "drugName"].value_counts())
drugname = list(rating.keys())
drug_rating = list(rating.values())

sns_rating = sns.barplot(x = drugname[0:20], y = drug_rating[0:20])

sns_rating.set_title('Top 20 drugs with 10/10 rating')
sns_rating.set_ylabel("Number of Ratings")
sns_rating.set_xlabel("Drug Names")
```

```
plt.setp(sns_rating.get_xticklabels(), rotation=90);
```

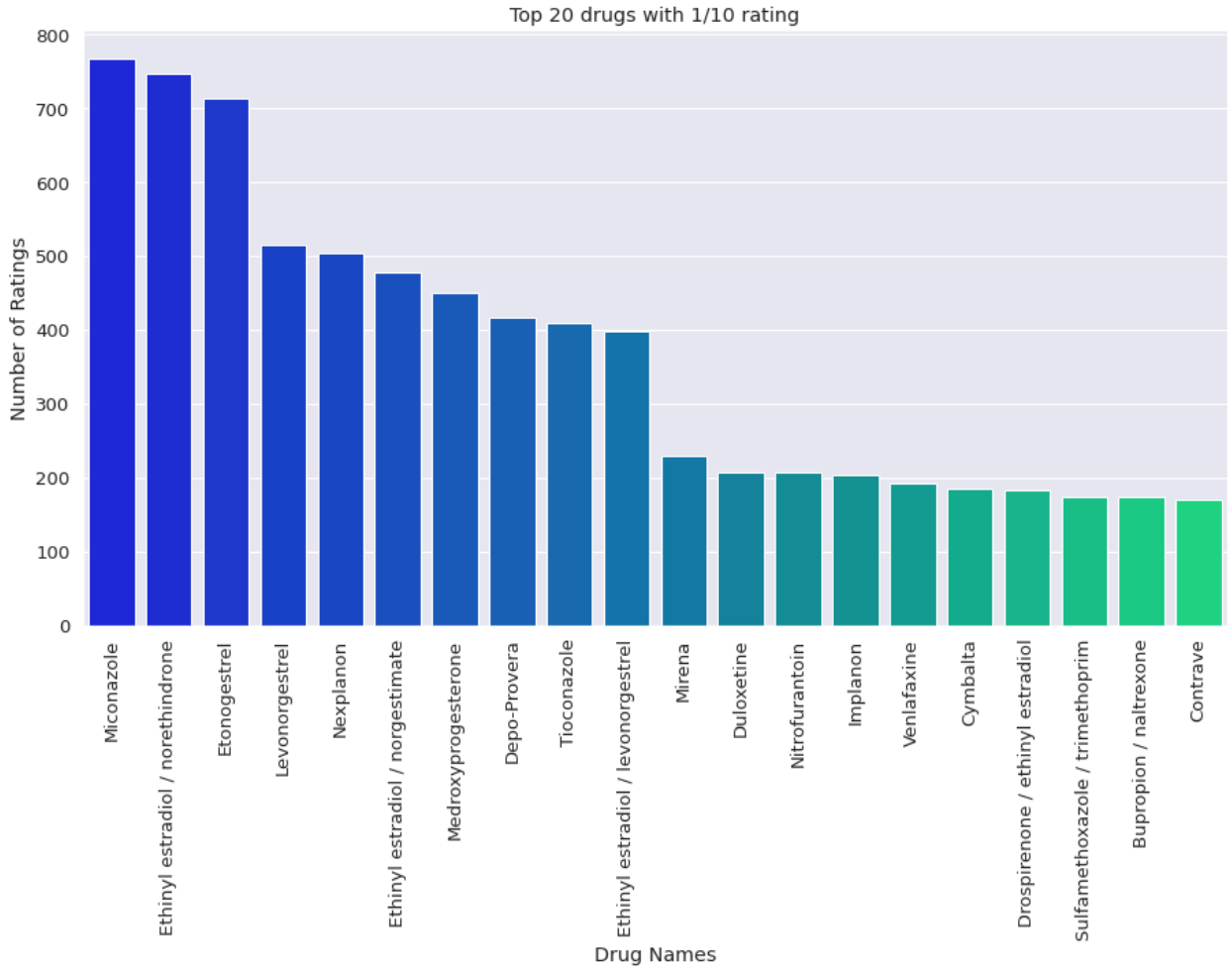


- The is a bar graph which shows the top 20 drugs given in the data set with a rating of 10/10. 'Levonorgestrel' is the drug with the highest number of 10/10 ratings, about 1883 Ratings in the data set for 'Levonorgestrel'.

In [10]:

```
# This barplot shows the Top 20 drugs with the 1/10 rating

# Setting the Parameter
sns.set(font_scale = 1.2, style = 'darkgrid')
plt.rcParams['figure.figsize'] = [15, 8]

rating = dict(data.loc[data.rating == 1, "drugName"].value_counts())
drugname = list(rating.keys())
drug_rating = list(rating.values())
```

```
sns_rating = sns.barplot(x = drugname[0:20], y = drug_rating[0:20],
palette = 'winter')

sns_rating.set_title('Top 20 drugs with 1/10 rating')
sns_rating.set_ylabel("Number of Ratings")
sns_rating.set_xlabel("Drug Names")
plt.setp(sns_rating.get_xticklabels(), rotation=90);
```

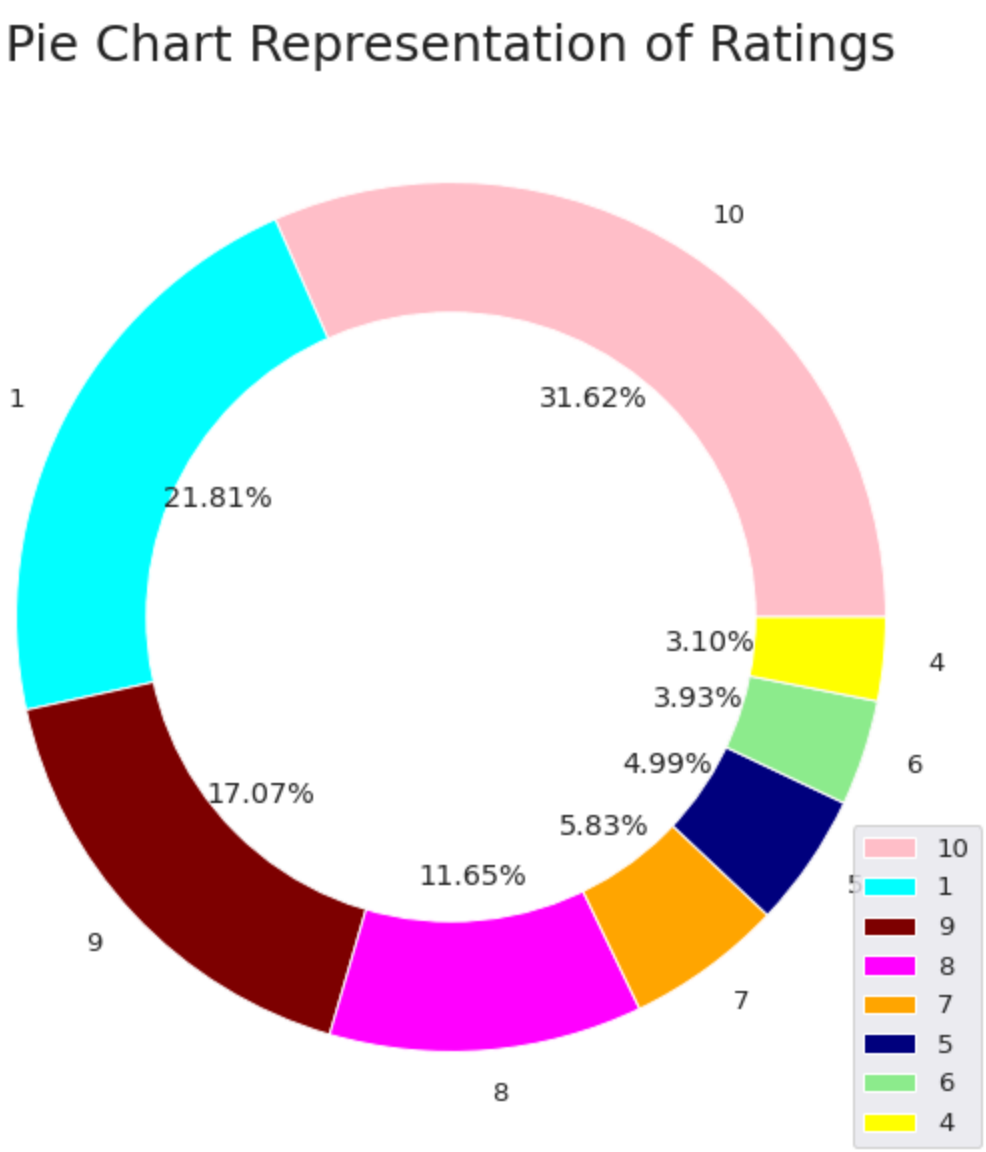


- The is a bar graph thatshows the top 20 drugs given in the data set with a rating of 1/10. 'Miconazole' is the drug with the highest number of 1/10 ratings, about 767.

In [11]:

```
# making a donut chart to represent share of each ratings

size = [68005, 46901, 36708, 25046, 12547, 10723, 8462, 6671]
```

```
colors = ['pink', 'cyan', 'maroon',  'magenta', 'orange', 'navy',
'lightgreen', 'yellow']
labels = "10", "1", "9", "8", "7", "5", "6", "4"

my_circle = plt.Circle((0, 0), 0.7, color = 'white')

plt.rcParams['figure.figsize'] = (10, 10)
plt.pie(size, colors = colors, labels = labels, autopct = '%.2f%%')
plt.axis('off')
plt.title('Pie Chart Representation of Ratings', fontsize = 25)
p = plt.gcf()
plt.gca().add_artist(my_circle)
plt.legend()
plt.show()
```



- This Pie Chart reprents the Rating of Reviews.

In [12]:

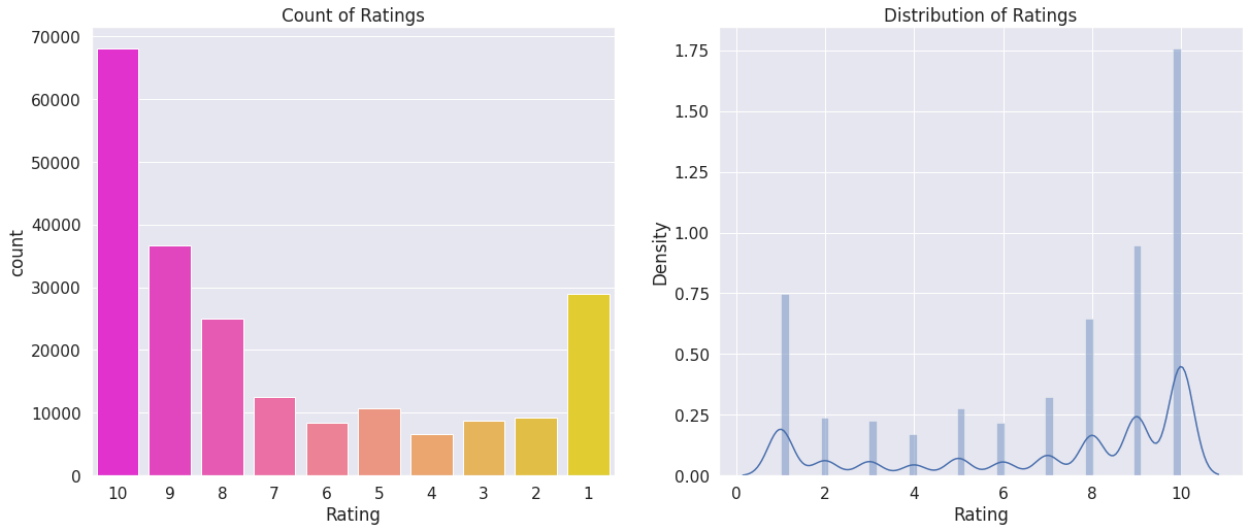
```
# A countplot of the ratings so we can see the distribution of the ratings
plt.rcParams['figure.figsize'] = [20,8]
sns.set(font_scale = 1.4, style = 'darkgrid')
fig, ax = plt.subplots(1, 2)

sns_1 = sns.countplot(data['rating'], palette = 'spring', order =
list(range(10, 0, -1)), ax = ax[0])
sns_2 = sns.distplot(data['rating'], ax = ax[1])
sns_1.set_title('Count of Ratings')
sns_1.set_xlabel("Rating")
```

```
sns_2.set_title('Distribution of Ratings')
sns_2.set_xlabel("Rating")
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in
a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[12]:

```
Text(0.5, 0, 'Rating')
```

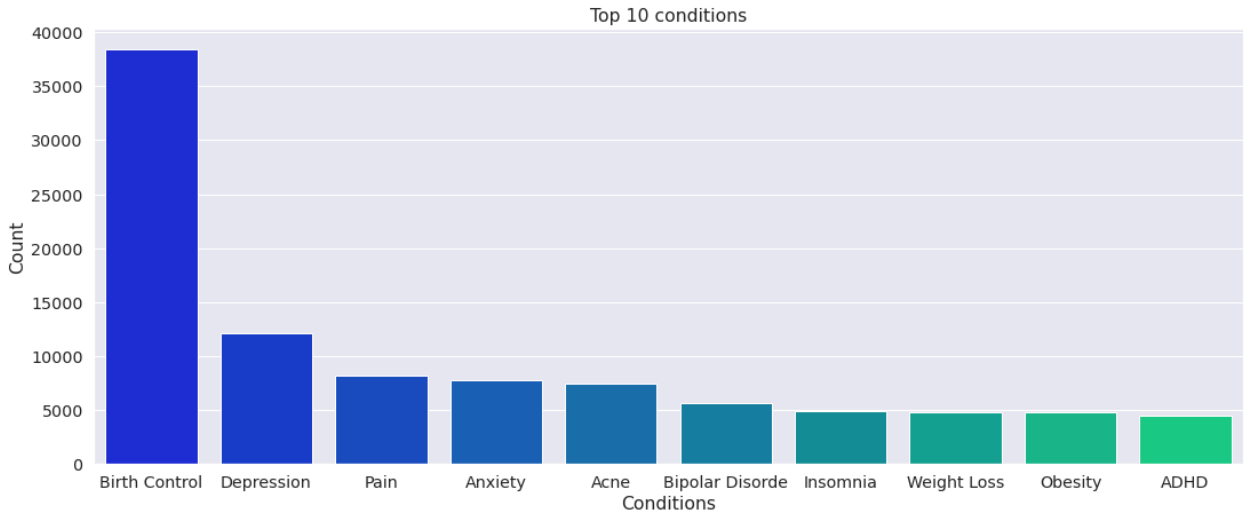


- The shows a distribution plot on the right hand side and a bar graph of the same on the left hand side. This shows the distribution of the ratings from 1 to 10 in the data set.

In [13]:

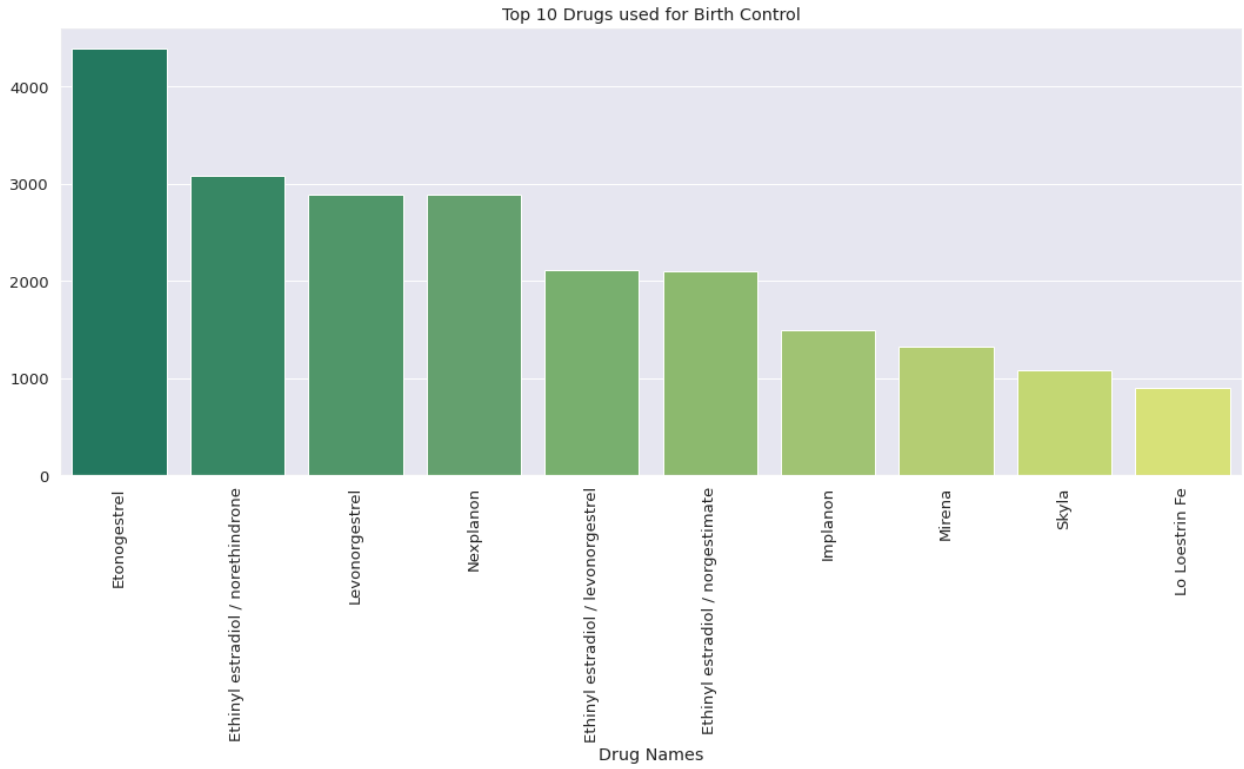
```
# This barplot show the top 10 conditions the people are suffering.
cond = dict(data['condition'].value_counts())
top_condition = list(cond.keys())[0:10]
```

```
values = list(cond.values())[0:10]
sns.set(style = 'darkgrid', font_scale = 1.3)
plt.rcParams['figure.figsize'] = [18, 7]

sns_ = sns.barplot(x = top_condition, y = values, palette = 'winter')
sns_.set_title("Top 10 conditions")
sns_.set_xlabel("Conditions")
sns_.set_ylabel("Count");
```



- The is a bar graph which exhibits the top 10 conditions the people are suffering from. In this data set 'Birth Control' is the most prominent condition by a very big margin followed by Depression and pain.

```
# Top 10 drugs which are used for the top condition, that is Birth Control
df1 = data[data['condition'] == 'Birth
Control']['drugName'].value_counts()[0: 10]
sns.set(font_scale = 1.2, style = 'darkgrid')

sns_ = sns.barplot(x = df1.index, y = df1.values, palette = 'summer')
sns_.set_xlabel('Drug Names')
sns_.set_title("Top 10 Drugs used for Birth Control")
plt.setp(sns_.get_xticklabels(), rotation = 90);
```



- The is a bar graph which exhibits the top 10 drug names for the people suffering from Birth Control. In this data set 'Etonogestrel' is the most prominent drug by a very big margin.

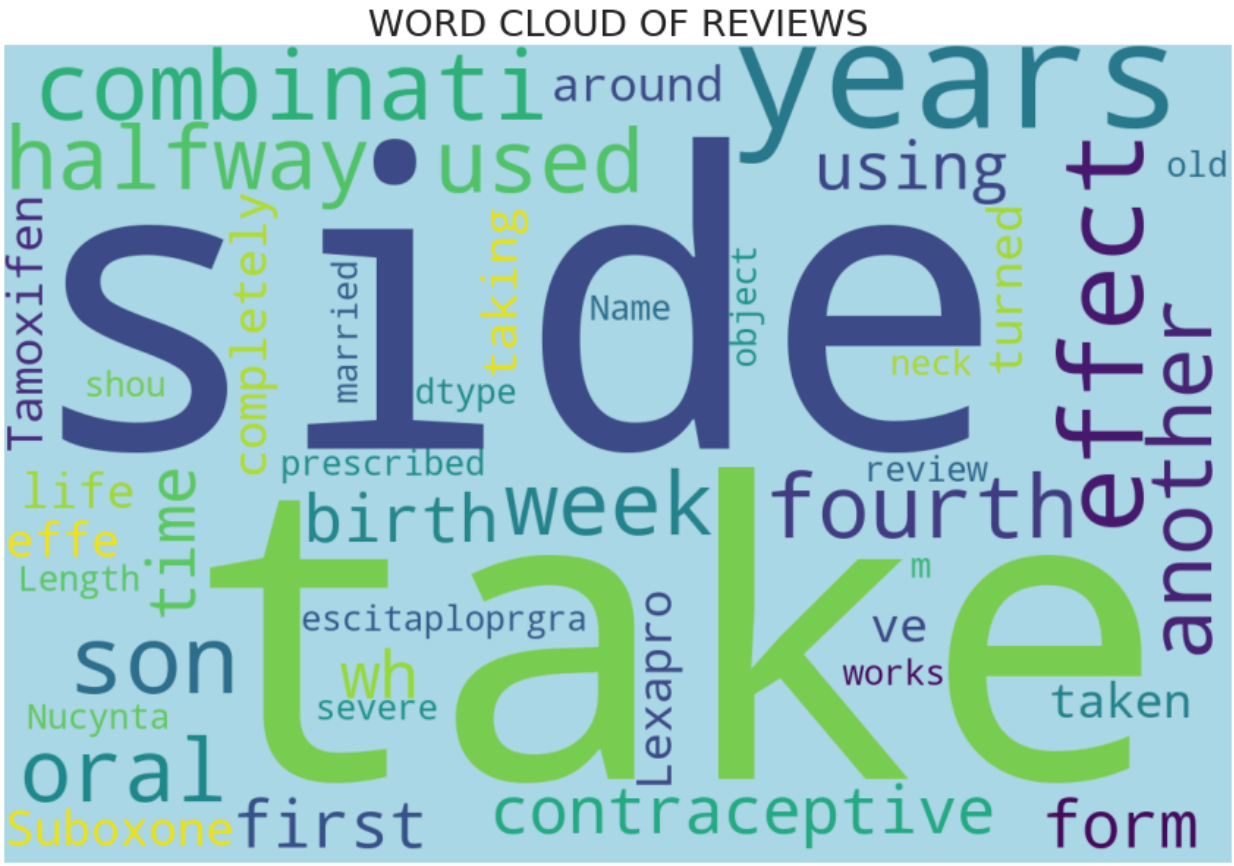
VISUALIZATION OF REVIEWS

In [15]:

```
# let's see the words cloud for the reviews

# most popular drugs

from wordcloud import WordCloud
from wordcloud import STOPWORDS

stopwords = set(STOPWORDS)

wordcloud = WordCloud(background_color = 'lightblue', stopwords =
stopwords, width = 1200, height = 800).generate(str(data['review']))
```

```
plt.rcParams['figure.figsize'] = (15, 15)
plt.title('WORD CLOUD OF REVIEWS', fontsize = 25)
print(wordcloud)
plt.axis('off')
plt.imshow(wordcloud)
plt.show()
```

```
<wordcloud.wordcloud.WordCloud object at 0x7f7cf943a9d0>
```



- This is a word cloud for the reviews.

In [16]:

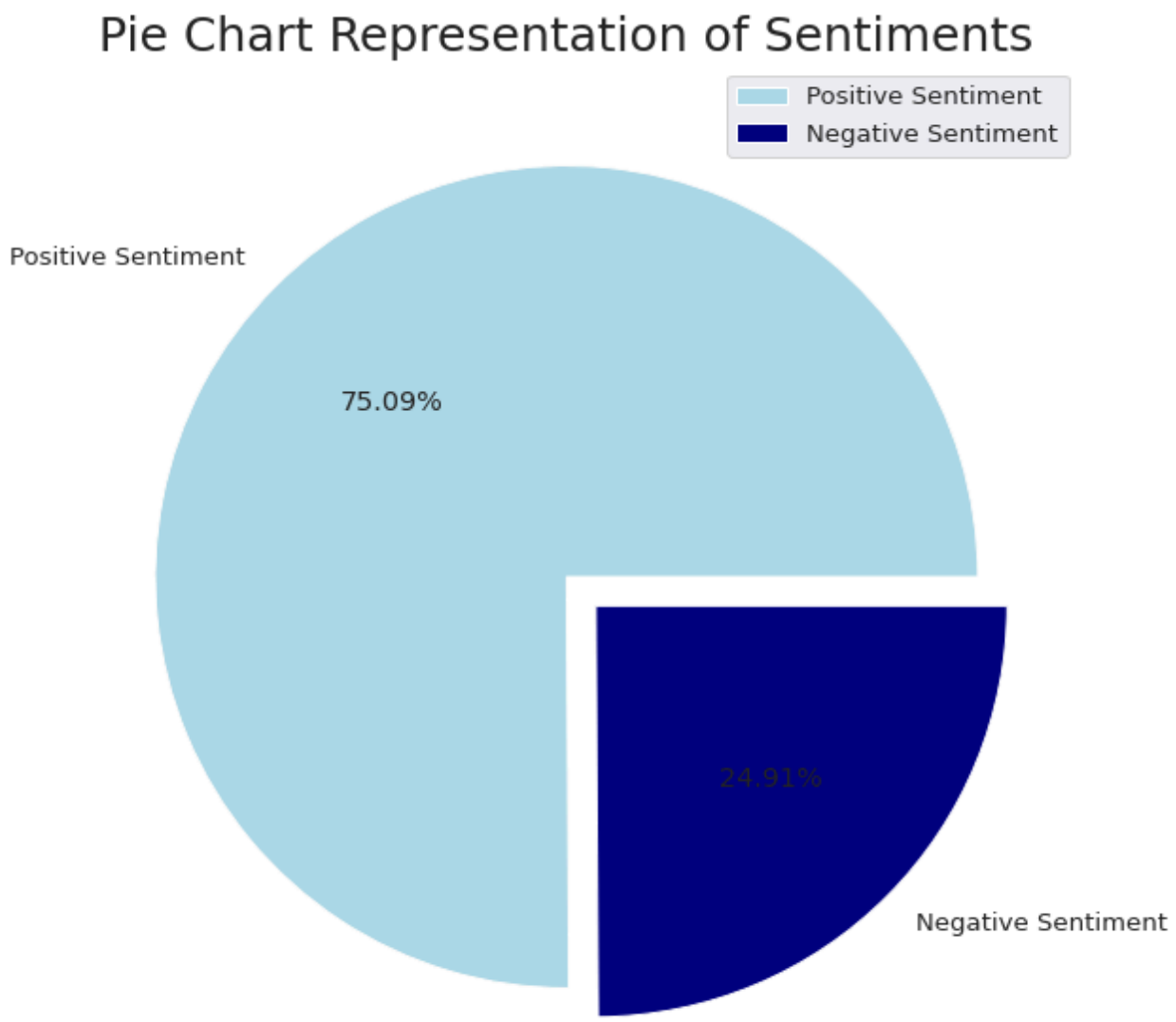
```
# feature engineering
# let's make a new column review sentiment

data.loc[(data['rating'] >= 5), 'Review_Sentiment'] = 1
data.loc[(data['rating'] < 5), 'Review_Sentiment'] = 0

data['Review_Sentiment'].value_counts()
```

Out[16]:

```
1.0    161491
0.0     53572
Name: Review_Sentiment, dtype: int64
```

In [17]:

```
# a pie chart to represent the sentiments of the patients

size = [161491, 53572]
colors = ['lightblue', 'navy']
labels = "Positive Sentiment","Negative Sentiment"
explode = [0, 0.1]

plt.rcParams['figure.figsize'] = (10, 10)
plt.pie(size, colors = colors, labels = labels, explode = explode, autopct
= '%.2f%%')
plt.axis('off')
plt.title('Pie Chart Representation of Sentiments', fontsize = 25)
plt.legend()
plt.show()
```



- This Pie Chart represents the Sentiments of the Reviews.

In [18]:

```
# making Words cloud for the postive sentiments

positive_sentiments = " ".join([text for text in
data['review'][data['Review_Sentiment'] == 1]])

from wordcloud import WordCloud
from wordcloud import STOPWORDS

stopwords = set(STOPWORDS)
wordcloud = WordCloud(background_color = 'magenta', stopwords = stopwords,
width = 1200, height = 800).generate(positive_sentiments)
```

```
plt.rcParams['figure.figsize'] = (15, 15)
plt.title('Word Cloud of Positive Reviews', fontsize = 30)
print(wordcloud)
plt.axis('off')
plt.imshow(wordcloud)
plt.show()
```

```
<wordcloud.wordcloud.WordCloud object at 0x7f7cf04d3810>
```

- This Pie Chart represents the Sentiments of the Reviews.

In [18]:

```
# making Words cloud for the postive sentiments

positive_sentiments = " ".join([text for text in
data['review'][data['Review_Sentiment'] == 1]])

from wordcloud import WordCloud
from wordcloud import STOPWORDS
```

```
stopwords = set(STOPWORDS)
wordcloud = WordCloud(background_color = 'magenta', stopwords = stopwords,
width = 1200, height = 800).generate(positive_sentiments)

plt.rcParams['figure.figsize'] = (15, 15)
plt.title('Word Cloud of Positive Reviews', fontsize = 30)
print(wordcloud)
plt.axis('off')
plt.imshow(wordcloud)
plt.show()
```

```
<wordcloud.wordcloud.WordCloud object at 0x7f7cf04d3810>
```

- This is a word cloud for the Negative Sentiments.

VISUALIZATION OF REVIEWS BASED ON DATETIME

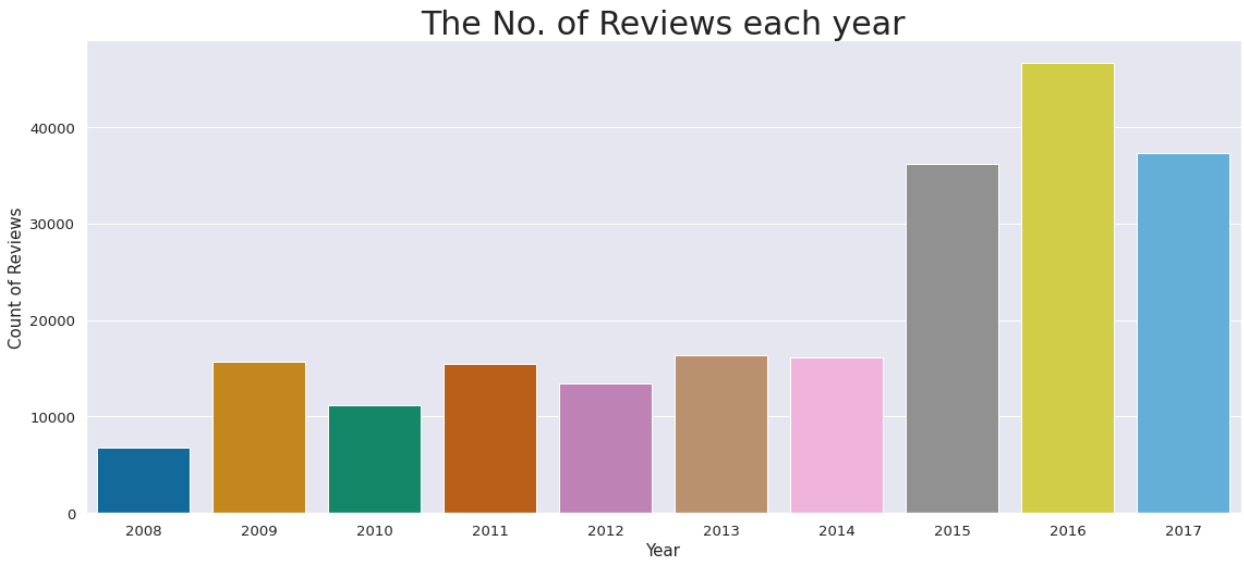
In [20]:

```
# converting the date into datetime format
data['date'] = pd.to_datetime(data['date'], errors = 'coerce')

# now extracting year from date
data['Year'] = data['date'].dt.year

# extracting the month from the date
data['month'] = data['date'].dt.month

# extracting the days from the date
data['day'] = data['date'].dt.day
```

In [21]:

```
# looking at the no. of reviews in each of the year

plt.rcParams['figure.figsize'] = (19, 8)
sns.countplot(data['Year'], palette ='colorblind')
plt.title('The No. of Reviews each year', fontsize = 30)
plt.xlabel('Year', fontsize = 15)
plt.ylabel('Count of Reviews', fontsize = 15)
plt.show()
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

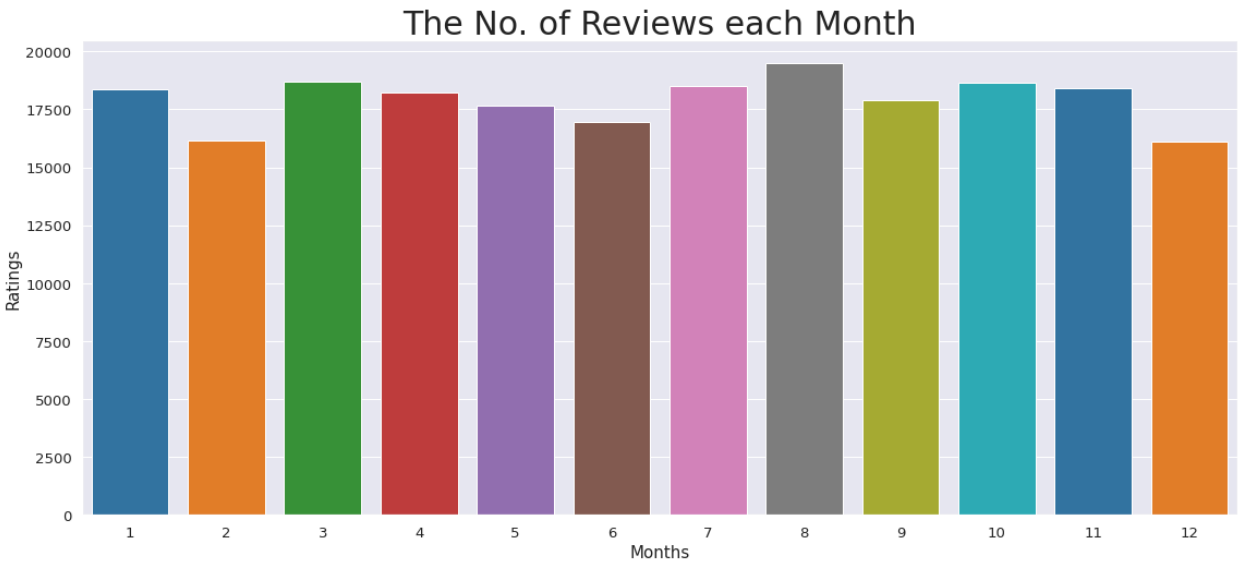


- The is a Bar graph that shows the number of reviews in the data set per year. It can be inferred that most ratings are given in 2016 and 2008 has the least number of reviews.

In [22]:

```
# looking at the no. of reviews in each of the months

plt.rcParams['figure.figsize'] = (19, 8)
sns.countplot(data['month'], palette ='tab10')
plt.title('The No. of Reviews each Month', fontsize = 30)
plt.xlabel('Months', fontsize = 15)
plt.ylabel('Ratings', fontsize = 15)
plt.show()
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

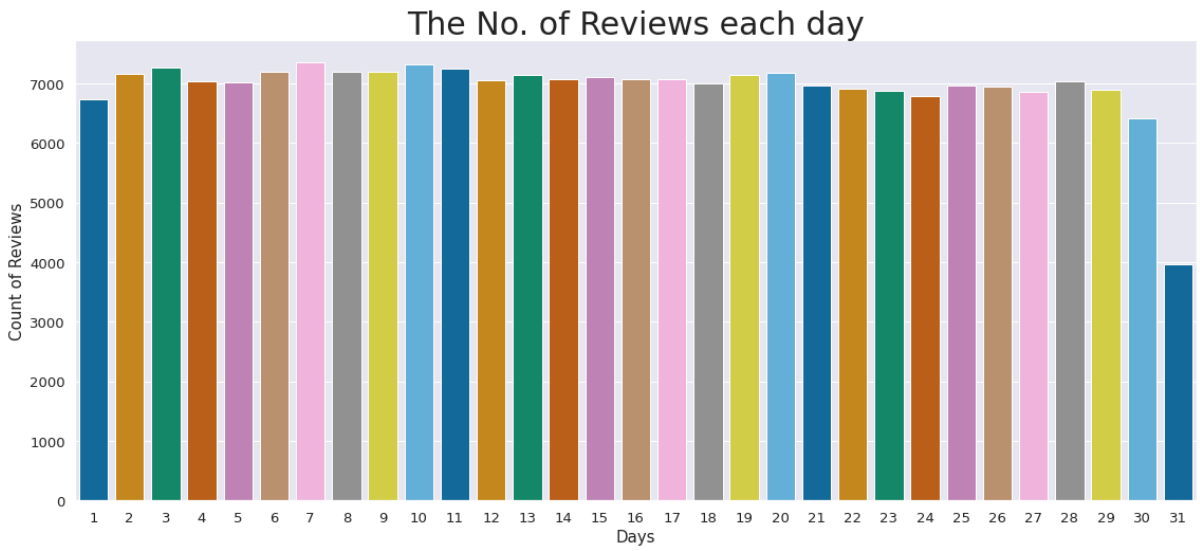


- The is a Bar graph that shows the numbe r of reviews in the data set per month.

- In [23]:

```
# looking at the no. of reviews in each of the day

plt.rcParams['figure.figsize'] = (19, 8)
sns.countplot(data['day'], palette ='colorblind')
plt.title('The No. of Reviews each day', fontsize = 30)
plt.xlabel('Days', fontsize = 15)
plt.ylabel('Count of Reviews', fontsize = 15)
plt.show()
```

- /opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning

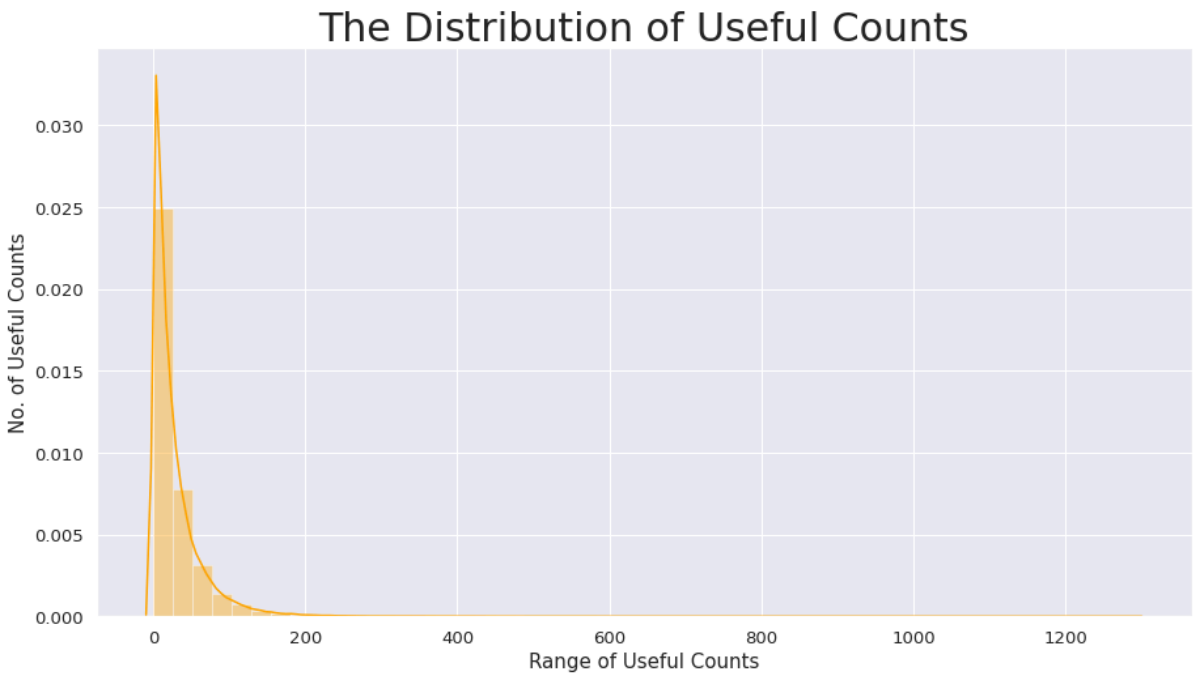


- The is a Bar graph that shows the number of reviews in the data set per day.
- ***VISUALIZATION OF USEFUL COUNT***

In [24]:

```
# plotting a dist plot

plt.rcParams['figure.figsize'] = (15, 8)
sns.distplot(data['usefulCount'], color = 'orange')
plt.title('The Distribution of Useful Counts', fontsize = 30)
plt.xlabel('Range of Useful Counts', fontsize = 15)
plt.ylabel('No. of Useful Counts', fontsize = 15)
plt.show()
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2557
: FutureWarning: `distplot` is a deprecated function and will be
removed in a future version. Please adapt your code to use either
`displot` (a figure-level function with similar flexibility) or
`histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



- This shows the distribution of the useful Counts in the data set.

DATA PREPROCESSING / FEATURE ENGINEERING

In [25]:

```
def review_clean(review):
    # changing to lower case
    lower = review.str.lower()

    # Replacing the repeating pattern of &#039;
    pattern_remove = lower.str.replace("&#039;", "")
```

```
    # Removing all the special Characters
    special_remove = pattern_remove.str.replace(r'[^\w\d\s]',' ')

    # Removing all the non ASCII characters
    ascii_remove = special_remove.str.replace(r'[^\x00-\x7F]+',' ')

    # Removing the leading and trailing Whitespaces
    whitespace_remove = ascii_remove.str.replace(r'^\s+|\s+?$','')

    # Replacing multiple Spaces with Single Space
    multiw_remove = whitespace_remove.str.replace(r'\s+',' ')

    # Replacing Two or more dots with one
    dataframe = multiw_remove.str.replace(r'\.{2,}', ' ')
```

```
    return dataframe

data['review_clean'] = review_clean(data['review'])
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:9:
FutureWarning: The default value of regex will change from True to False
in a future version.

  if __name__ == '__main__':

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:12:
FutureWarning: The default value of regex will change from True to False
in a future version.

  if sys.path[0] == '':
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:15:
FutureWarning: The default value of regex will change from True to False
in a future version.

  from ipykernel import kernelapp as app

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:18:
FutureWarning: The default value of regex will change from True to False
in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:21:
FutureWarning: The default value of regex will change from True to False
in a future version.
```

In [27]:

```
from textblob import TextBlob

from nltk.corpus import stopwords

from collections import Counter

import warnings; warnings.simplefilter('ignore')

import nltk

import string
```

```
from nltk import ngrams

from nltk.tokenize import word_tokenize

from nltk.stem import SnowballStemmer



# Removing the stopwords

stop_words = set(stopwords.words('english'))

data['review_clean'] = data['review_clean'].apply(lambda x: ' '.join(word
for word in x.split() if word not in stop_words))
```

- I have used textblob module to give the sentiment polarity of the review. This polarity is given to both the cleaned and uncleaned review

In [28]:

```
# Removing the word stems using the Snowball Stemmer

Snow_ball = SnowballStemmer("english")

data['review_clean'] = data['review_clean'].apply(lambda x: "
".join(Snow_ball.stem(word) for word in x.split()))
```

In [29]:

```
data.head(3)
```

uniqueID	drugName	condition	review	rating	date	usefulCount	Review_Sentiment	Year	month	day	review_clean	
0	206461	Valsartan	Left Ventricular Dysfunction	"It has no side effect, I take it in combinati...	9	2012-05-20	27	1.0	2012	5	20	side effect take combin bystol 5 mg fish oil
1	95260	Guanfacine	ADHD	"My son is halfway through his fourth week of ...	8	2010-04-27	192	1.0	2010	4	27	son halfway fourth week intuniv becam concern ...
2	92703	Lybrel	Birth Control	"I used to take another oral contraceptive, wh...	5	2009-12-14	17	1.0	2009	12	14	use take anoth oral contracept 21 pill

												cycl ha

```
def sentiment(review):

    # Sentiment polarity of the reviews

    pol = []

    for i in review:

        analysis = TextBlob(i)

        pol.append(analysis.sentiment.polarity)

    return pol
```

In [31]:

```
data['sentiment'] = sentiment(data['review'])
```

In [32]:

```
data['sentiment_clean'] = sentiment(data['review_clean'])
```

In [33]:

```
# Cleaning the reviews without removing the stop words and using snowball
stemmer

data['review_clean_ss'] = review_clean(data['review'])

data['sentiment_clean_ss'] = sentiment(data['review_clean_ss'])
```

In [34]:

```
data = data.dropna(how="any", axis=0)
```

In [35]:

```
#Word count in each review

data['count_word']=data["review_clean_ss"].apply(lambda x:
len(str(x).split()))



#Unique word count

data['count_unique_word']=data["review_clean_ss"].apply(lambda x:
len(set(str(x).split())))
```

```
#Letter count

data['count_letters']=data["review_clean_ss"].apply(lambda x:
len(str(x)))

#punctuation count

data["count_punctuations"] = data["review"].apply(lambda x: len([c for c
in str(x) if c in string.punctuation]))



#upper case words count

data["count_words_upper"] = data["review"].apply(lambda x: len([w for w
in str(x).split() if w.isupper()]))



#title case words count

data["count_words_title"] = data["review"].apply(lambda x: len([w for w
in str(x).split() if w.istitle()]))



#Number of stopwords

data["count_stopwords"] = data["review"].apply(lambda x: len([w for w in
str(x).lower().split() if w in stop_words]))
```

```
#Average length of the words

data["mean_word_len"] = data["review_clean_ss"].apply(lambda x:
np.mean([len(w) for w in str(x).split()]))
```

- The new features engineered are 'count_word' which is the number of words in each review, 'count_unique_word' which is the number of the unique words in the reviews. 'count_letters' is the letter count, 'punctuation_count' is the punctuation count, 'count_words_upper' is the upper case word count,'count_words_title' is the title case word counts, 'count_stopwords' is the number of stop words in the review, and the 'mean_word_len' is the average length of the words in the review. The date is also divided into three columns which are day, month and year for separate features for training.

```
data.columns
```

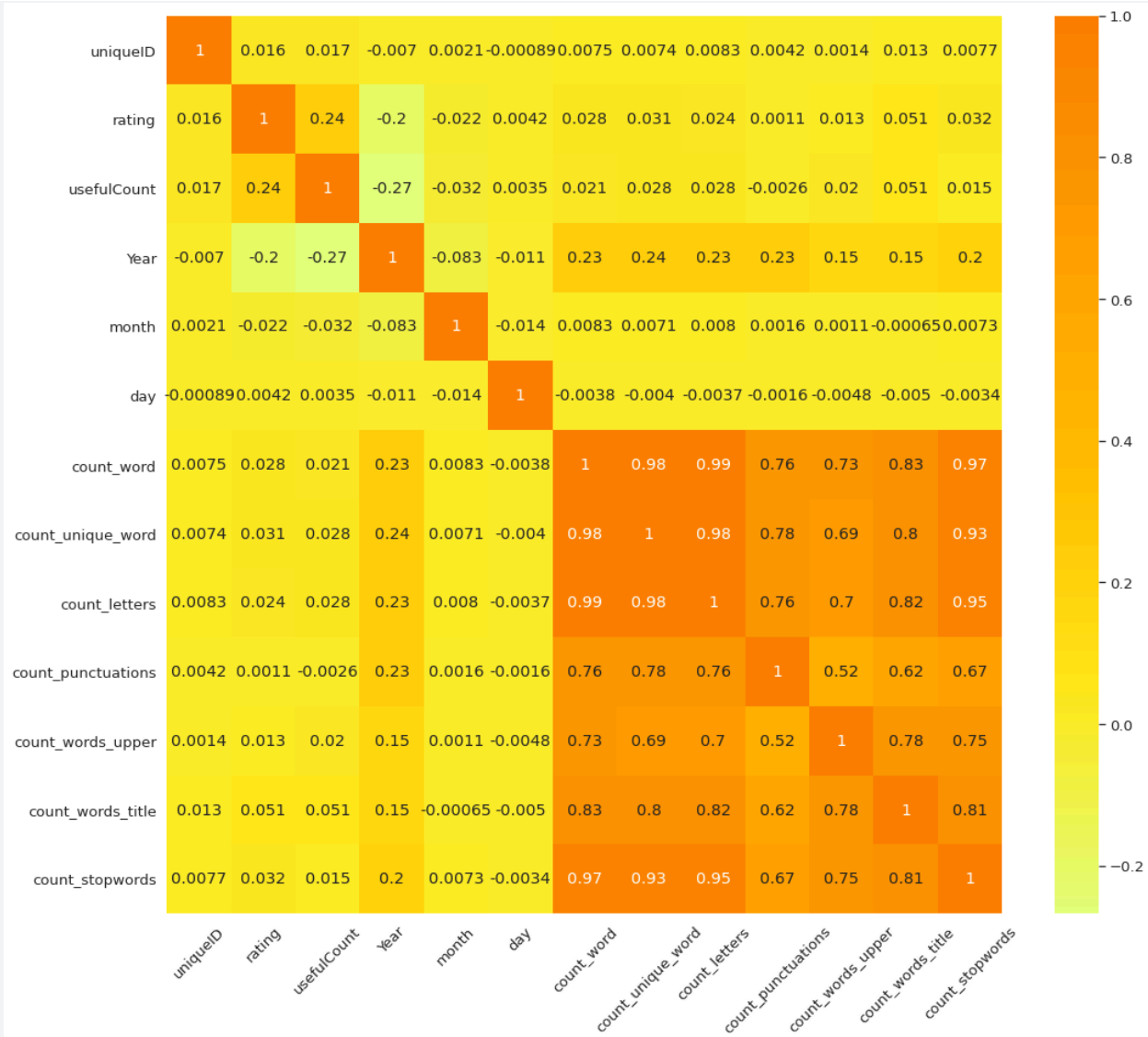
Out[36]:

```
Index(['uniqueID', 'drugName', 'condition', 'review', 'rating', 'date',
       'usefulCount', 'Review_Sentiment', 'Year', 'month', 'day',
       'review_clean', 'sentiment', 'sentiment_clean', 'review_clean_ss',
       'sentiment_clean_ss', 'count_word', 'count_unique_word',
       'count_letters', 'count_punctuations', 'count_words_upper',
       'count_words_title', 'count_stopwords', 'mean_word_len'],
      dtype='object')
```

CORRELATION MATRIX

In [37]:

```
# Correlation Heatmap of the features engineered

plt.rcParams['figure.figsize'] = [17,15]

sns.set(font_scale = 1.2)

corr = data.select_dtypes(include = 'int64').corr()

sns_ = sns.heatmap(corr, annot = True, cmap = 'Wistia')

plt.setp(sns_.get_xticklabels(), rotation = 45);
```



- Correlation Heatmap is plotted using seaborn which contains all the new features engineered and the old features.

LABEL ENCODING

In [38]:

```
# Label Encoding Drugname and Conditions

from sklearn.preprocessing import LabelEncoder

label_encoder_feat = {}

for feature in ['drugName', 'condition']:

    label_encoder_feat[feature] = LabelEncoder()

    data[feature] =
label_encoder_feat[feature].fit_transform(data[feature])
```

- The Label Encoder is used to change the categorical values of Drug Names and the conditions in to numerical values for the machine learning modelling. There are 3,667 unique drugs in the dataset that's why One hot encoder is not used as it would generate 3,667 new features and it would be very computationally expensive.

LIGHT GBM MODEL

- LightGBM is a gradient boosting framework that uses treebased learning algorithms. It's designed to be distributed and efficient. It has many advantages like faster training speed and higher efficiency, lower memory usage, better accuracy and support of parallel and GPU learning, since it is based on decision tree algorithms, it splits the tree leaf wise with the best fit.

In [39]:

```
# Importing Libraries for the Machine Learning Model

from xgboost import XGBClassifier

from lightgbm import LGBMModel,LGBMClassifier, plot_importance

from sklearn.metrics import confusion_matrix, accuracy_score,
classification_report

from sklearn.model_selection import train_test_split
```

In [40]:

```
# Defining Features and splitting the data as train and test set
```

```
features = data[['condition', 'usefulCount', 'sentiment', 'day', 'month',
'Year',
                 'sentiment_clean_ss', 'count_word',
'count_unique_word', 'count_letters',
                 'count_punctuations', 'count_words_upper',
'count_words_title',
                 'count_stopwords', 'mean_word_len']]


target = data['Review_Sentiment']


X_train, X_test, y_train, y_test = train_test_split(features, target,
test_size = 0.3, random_stat
e = 42)
print ("The Train set size ", X_train.shape)
print ("The Test set size ", X_test.shape)


The Train set size  (149708, 15)
```

```
The Test set size  (64161, 15)
```

- 70% of the dataset is used for the training and the rest of the data i.e. 30% is used for the testing purpose. The shape of the training set is (149708, 15) and the shape of the test set is (64161, 15).

In [41]:

```
# Training Model - I

clf = LGBMClassifier(

        n_estimators=10000,

        learning_rate=0.10,

        num_leaves=30,

        subsample=.9,

        max_depth=7,

        reg_alpha=.1,

        reg_lambda=.1,
```

```
        min_split_gain=.01,

        min_child_weight=2,

        silent=-1,

        verbose=-1,

        )

model = clf.fit(X_train, y_train)

# Predictions

predictions = model.predict(X_test)

print ("The Accuracy of the model is : ", accuracy_score(y_test,
predictions))

print ("The confusion Matrix is ")

confusion_matrix(y_test, predictions)
```

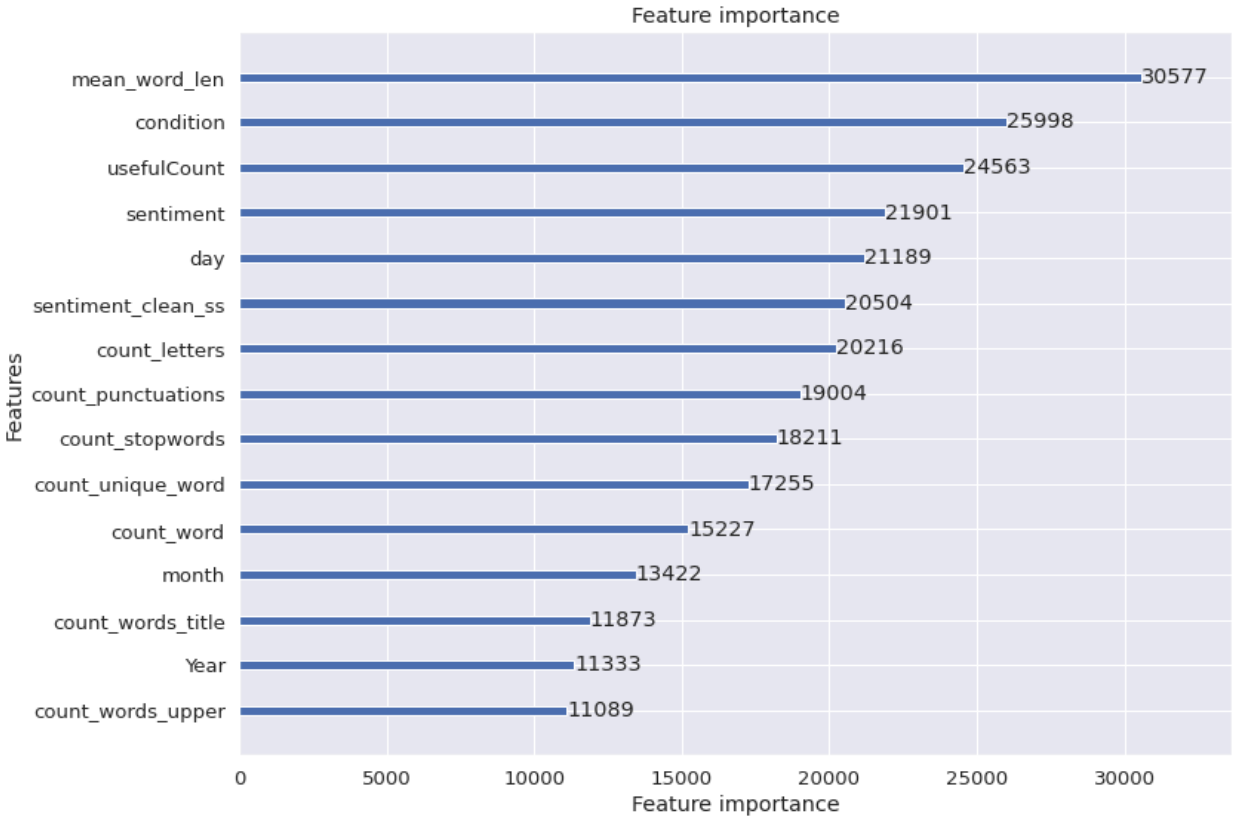
```
The Accuracy of the model is :  0.9014977946104331

The confusion Matrix is
```

Out[41]:

```
array([[11753,  4321],
       [ 1999, 46088]])
```

- The Confusion Matrix for the LGBM model is given above, it can be seen that the accuracy of the LGBM is 0.9014 (90%).

```
# Feature Importance Plot using LGBM

plt.rcParams['figure.figsize'] = [12, 9]

sns.set(style = 'darkgrid', font_scale = 1.2)

plot_importance(model);
```



- Above figure depicts the feature importance plot using the LightGBM. It can be inferred that the most importance feature is the mean word length and after that the condition of the patient. The least important feature of them all is the upper-case word count.