Team Members

Aysenur Yilmaz
aysenuryilmaz95@gmail.com

Mustafa Abdullah Hakkoz
hakkoz@gmail.com

DRUG REVIEWS CLASSIFIER

Can you predict the patient's condition based on the review?

An end-to-end Machine Learning Project

1. Introduction

The main goal of this project is to show a sample case study for machine learning by building an end-to-end application that can be used and found beneficial in real life.

Basically, we have tried to create a real-time web application that takes input text from the user and predicts its diagnosis out of 10 predefined labels.

In this project, this is achieved by supervised machine learning classification models, we have used six different classical machine learning classifiers: SVM, Logistic Regression, Naive Bayes and also some Boosting algorithms: Random Forest and XGBoost. Moreover, we have built a convolutional neural network. One of the objectives of this project is comparing the algorithms mentioned above.

This is a text classification problem and it is one of the most popular NLP applications.

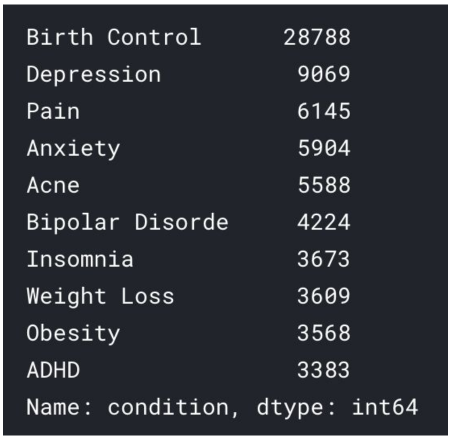
In this project, you will see an entire end-to-end machine learning project from beginning to the end.

2. Dataset

In this project, UCI ML Drug Review dataset is used. For dataset, please follow the link: https://www.kaggle.com/jessicali9530/kuc-hackathon-winter-2018

Train set consists of 161297 samples and the test set consists of 53766 samples of drug reviews.

There are 885 different conditions in the train set. Also, the test set has 709 different conditions. Then I decided to use top 10 conditions.

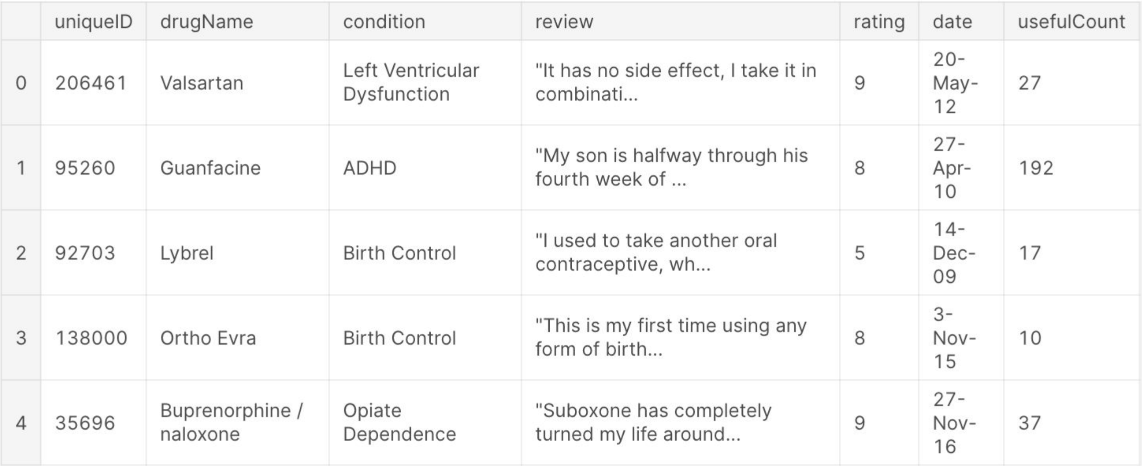


```
Birth Control      28788
Depression          9069
Pain                6145
Anxiety             5904
Acne                5588
Bipolar Disorde     4224
Insomnia            3673
Weight Loss         3609
Obesity             3568
ADHD                3383
Name: condition, dtype: int64
```

3. Methodology

3.1. Exploratory Data Analysis

Exploratory Data Analysis also known as EDA is a popular practice for gaining insights about the data. We have examined the dataset using some plots and tables.



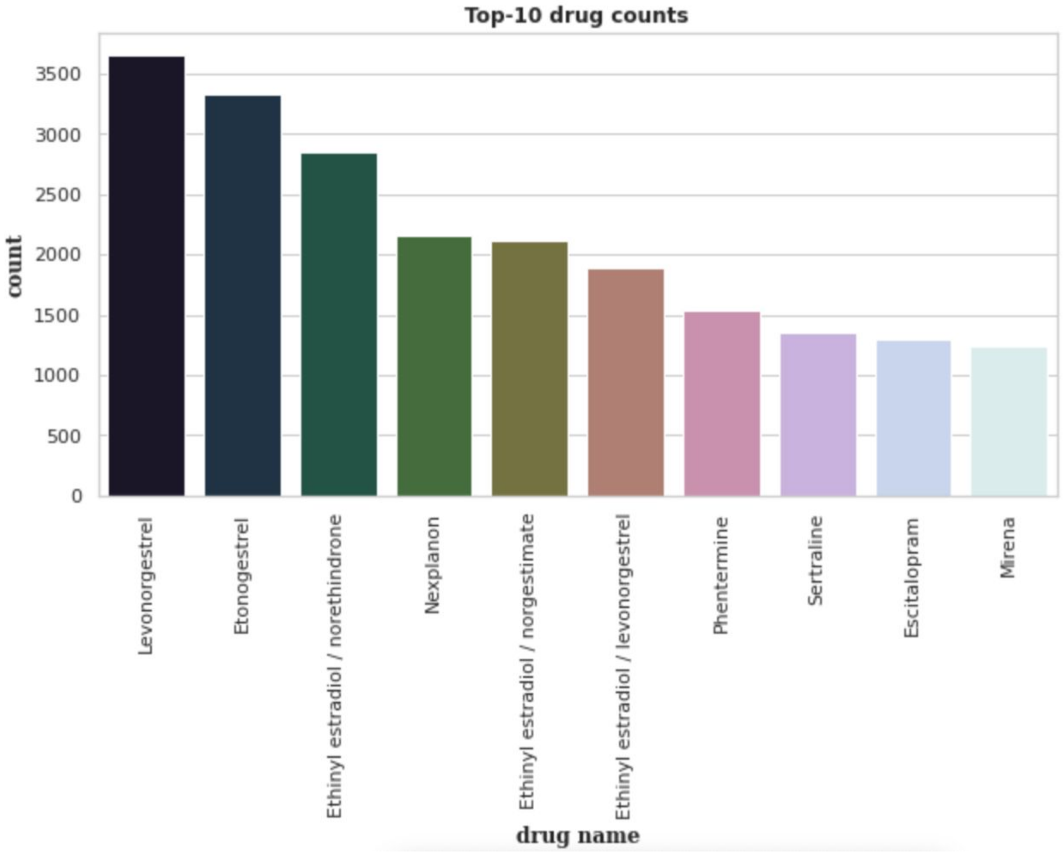
	uniqueID	drugName	condition	review	rating	date	usefulCount
0	206461	Valsartan	Left Ventricular Dysfunction	"It has no side effect, I take it in combinati...	9	20-May-12	27
1	95260	Guanfacine	ADHD	"My son is halfway through his fourth week of ...	8	27-Apr-10	192
2	92703	Lybrel	Birth Control	"I used to 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

You can see the first five samples of the dataset, there are seven columns:

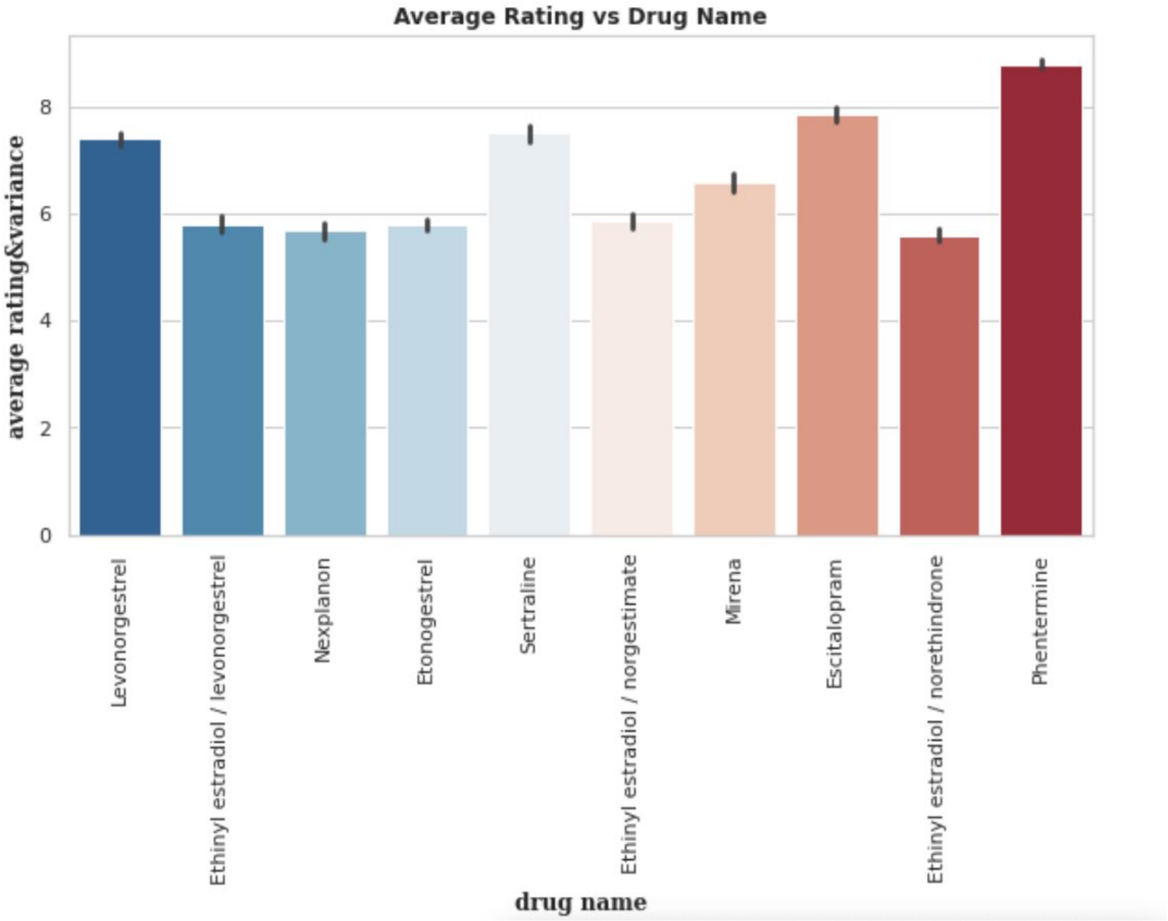
- uniqueID: belongs to review
- drugName: name of the drug the review is written for
- condition: diagnosis of the review
- rating: rating of the user for a specific drug taken under specific condition
- date: belongs to review
- usefulCount: rating of the review given by other users

It is possible to use drug name, condition or rating as target label however we chose "condition" since we decided to detect the topic of a review. Drug names would be a bad choice because it could be used in review text also. Plus, choosing rating as a label would be a regression problem. After selecting top-10 conditions train set size is reduced to 73951 from 161297 and test set is reduced 24772 from 53766 samples.

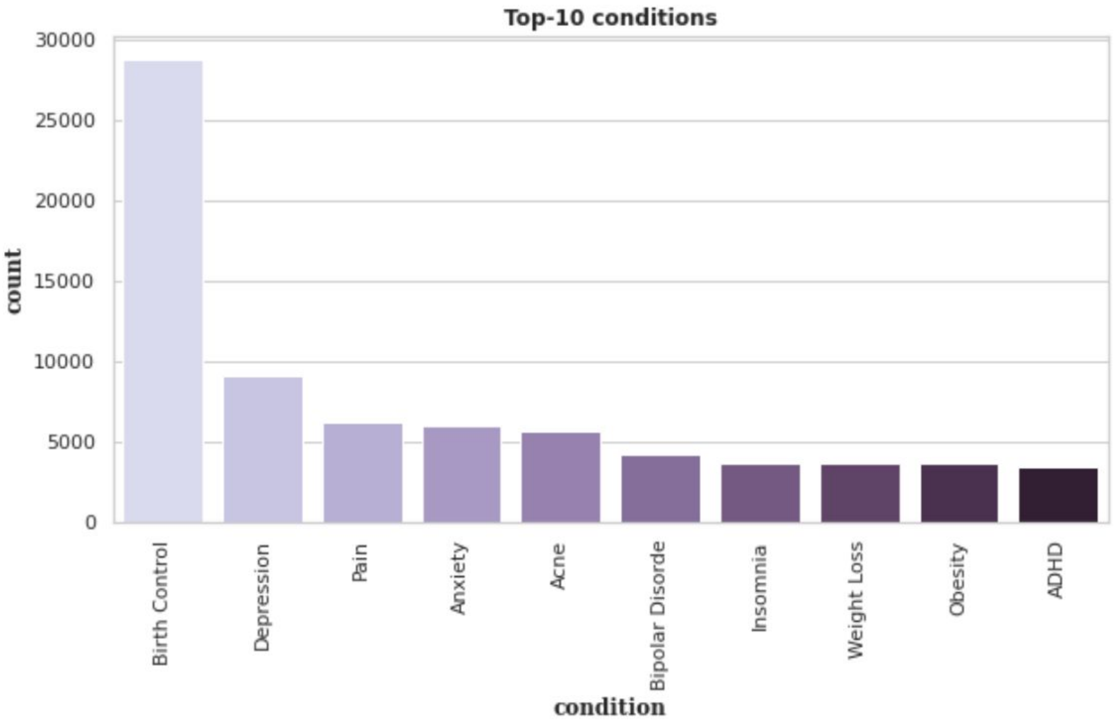
You can see the review counts for top-10 drugs. They are not equally balanced.



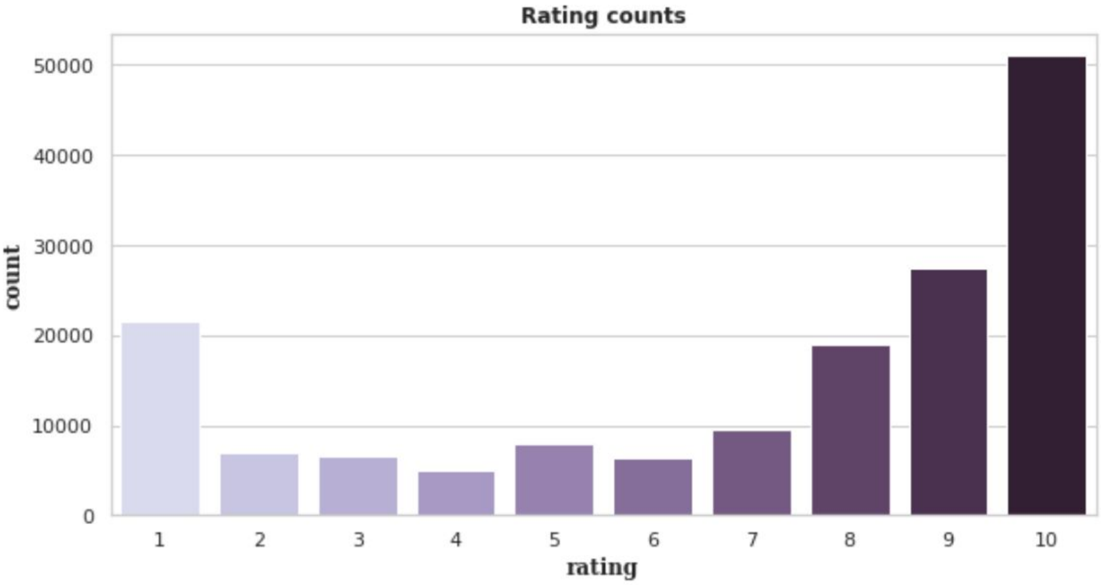
Below, you can also see average ratings of top-10 drugs. It could be seen that ratings and review counts are correlated for popular drugs.



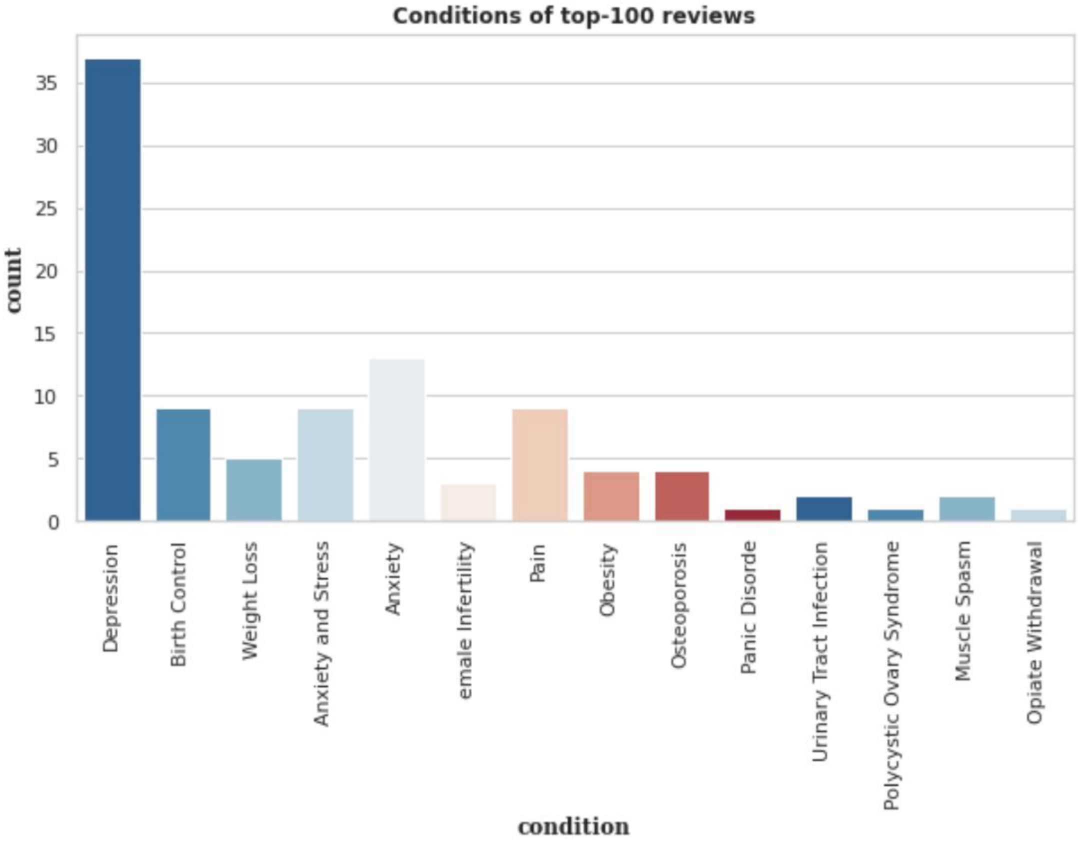
You could see the numbers of top-10 conditions. Except “birth control”, rest of conditions are almost balanced.



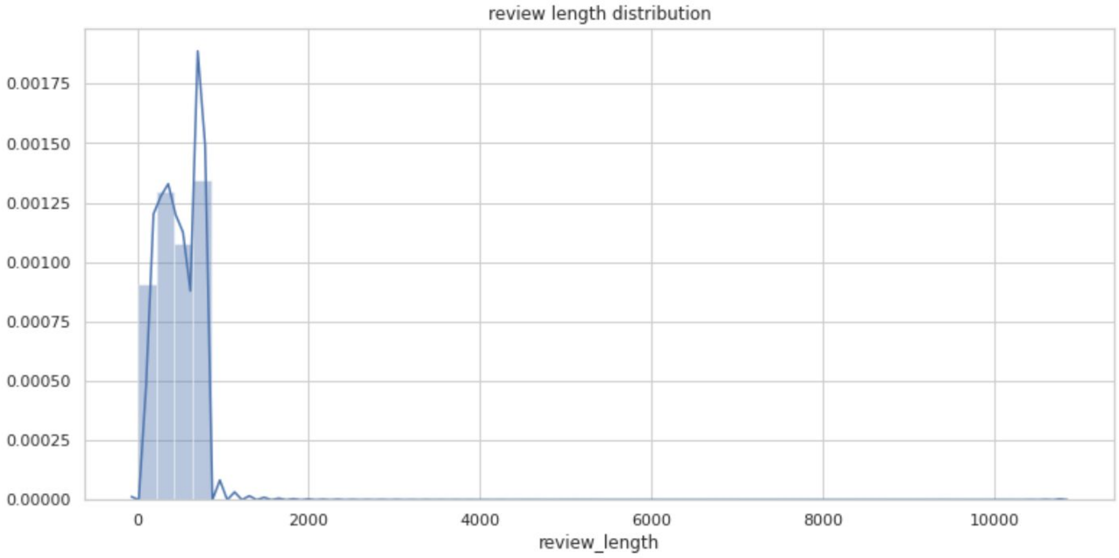
As you can see generally people tend to give the highest or lowest rating and ignore the ratings between 2 and 7.



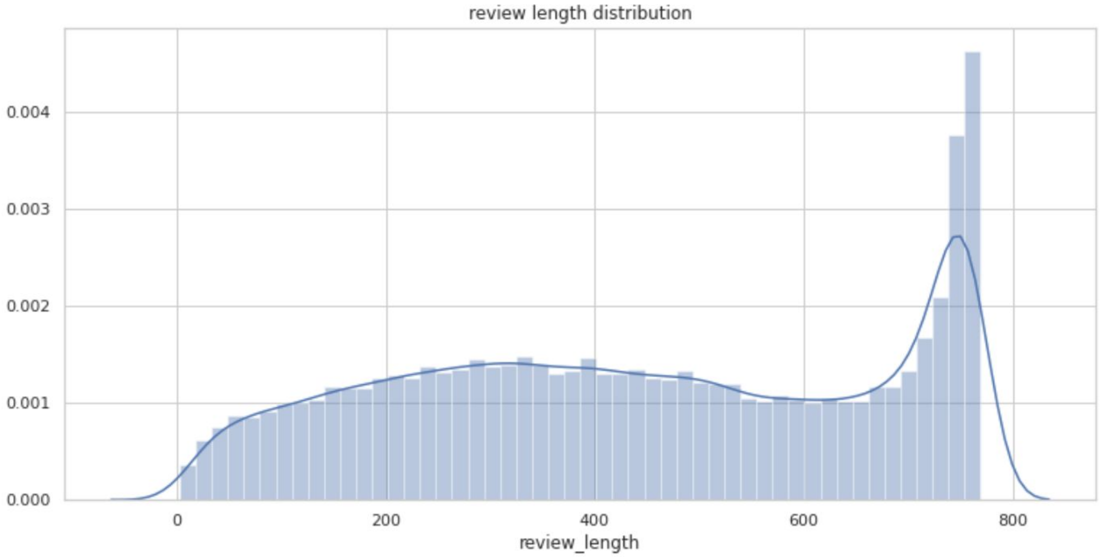
Here, top-100 reviews selected according to their “usefulCount” numbers. As can be seen “depression” has the most popular condition amongst them.



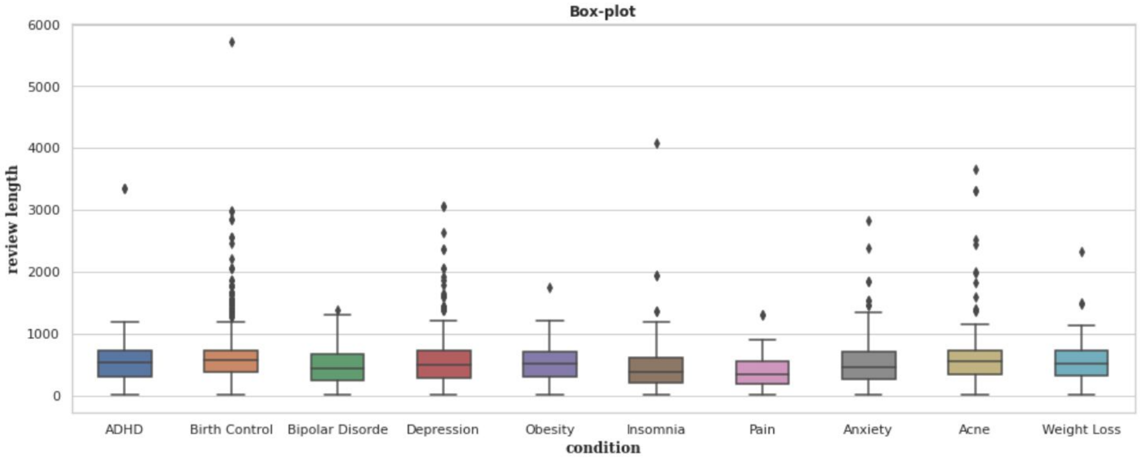
This is the density distribution of review length. Since it is not readable well, outliers are removed.



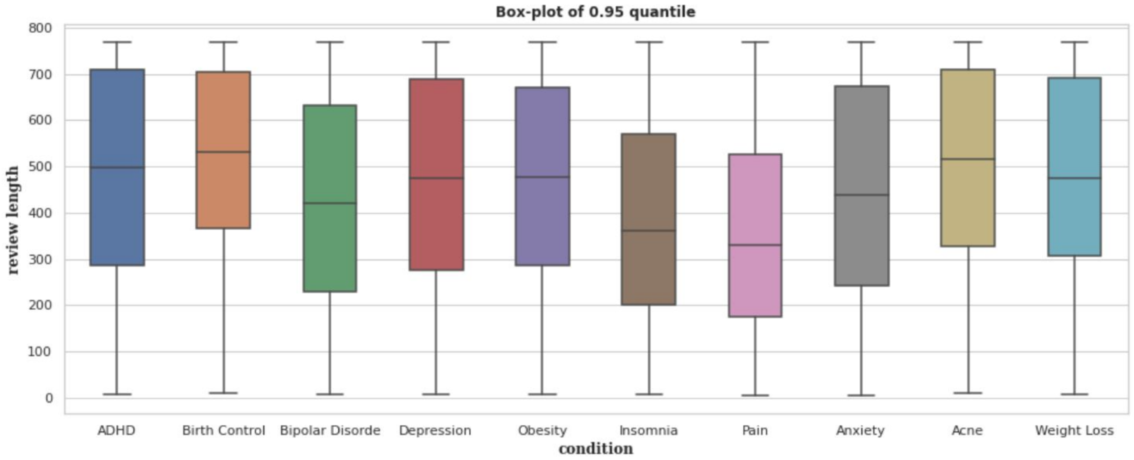
Here is the same plot with more details. For some review lengths that are between 740 - 770 have the most densities and the rest are equally balanced.



You can see the outlier analysis of the dataset with box-plot. **Birth control, depression , anxiety** and **acne** have a significant amount of outliers. Hence they are removed to see more details.



You could see a clear version of this box-plot below. Their distribution and variances are very similar.

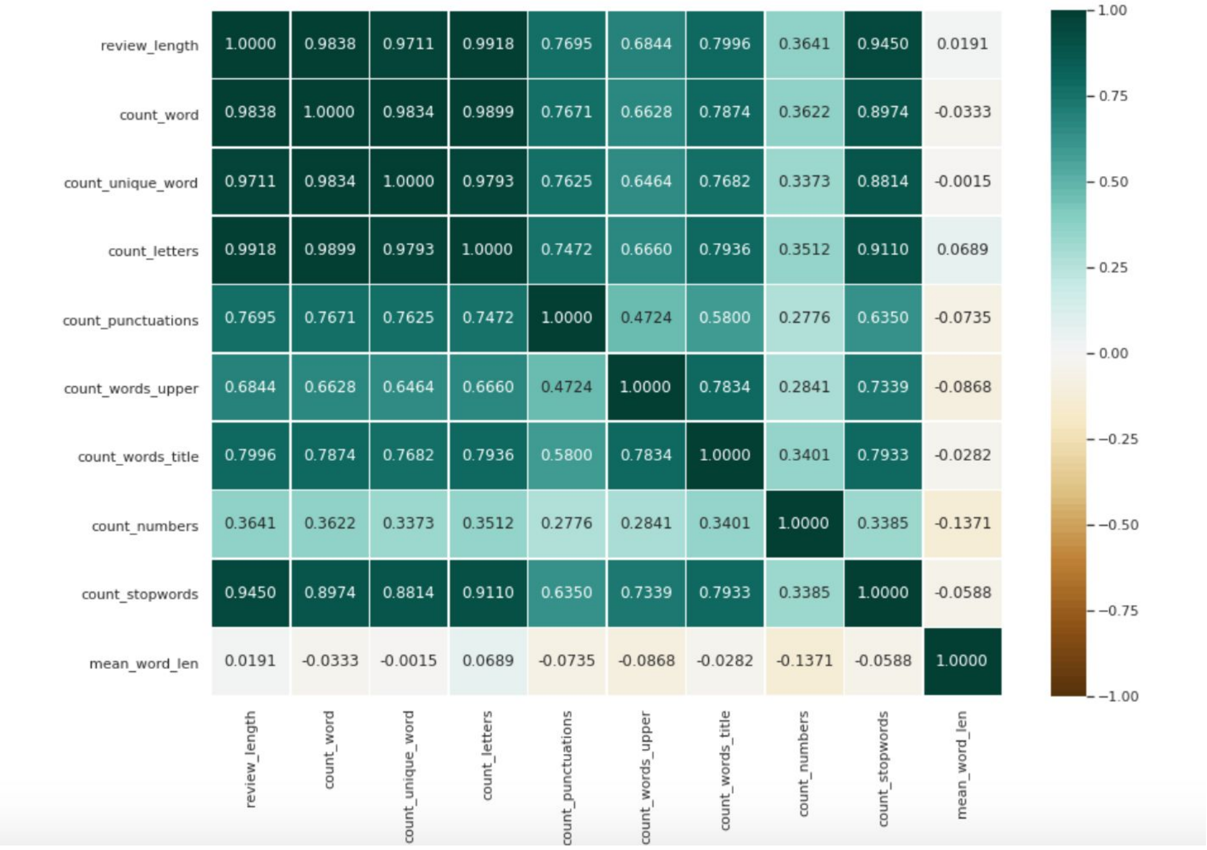


3.2. Feature Engineering

For this notebook, steps are listed below:

- Text Cleaning and Preparation
- Label Encoding
- Train-test split

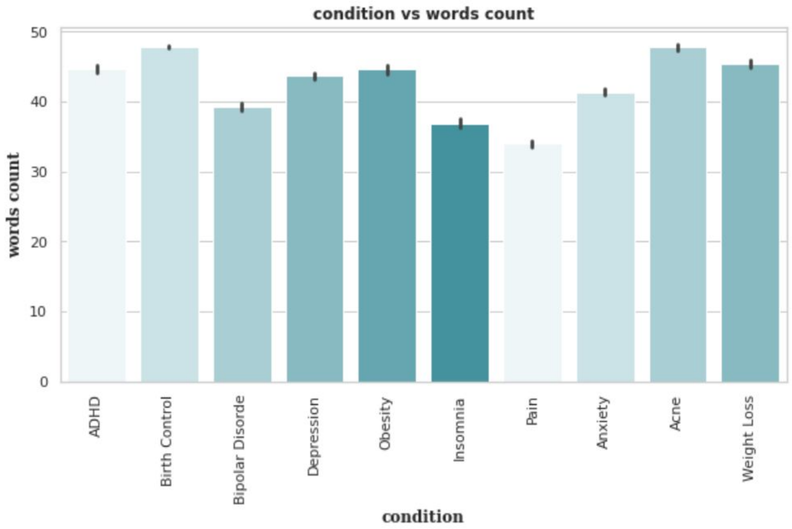
We removed special characters, punctuation signs, numbers, possessive pronouns in order to clean the text. After that, all of the letters are converted to lowercase form. For the lemmatization step, we used WordNetLemmatizer() and built-in stopwords of nltk library.



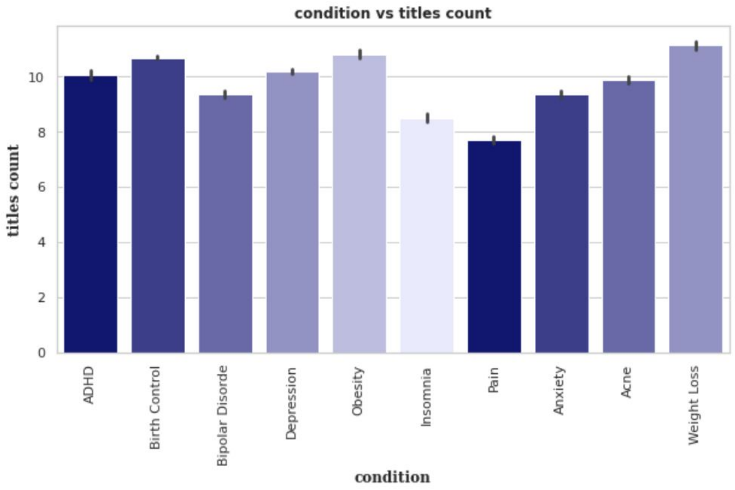
	review_length	count_word	count_unique_word	count_letters	count_punctuations	count_words_upper	count_words_title	count_numbers	count_stopwords	mean_word_len
review_length	1.0000	0.9838	0.9711	0.9918	0.7695	0.6844	0.7996	0.3641	0.9450	0.0191
count_word	0.9838	1.0000	0.9834	0.9899	0.7671	0.6628	0.7874	0.3622	0.8974	-0.0333
count_unique_word	0.9711	0.9834	1.0000	0.9793	0.7625	0.6464	0.7682	0.3373	0.8814	-0.0015
count_letters	0.9918	0.9899	0.9793	1.0000	0.7472	0.6660	0.7936	0.3512	0.9110	0.0689
count_punctuations	0.7695	0.7671	0.7625	0.7472	1.0000	0.4724	0.5800	0.2776	0.6350	-0.0735
count_words_upper	0.6844	0.6628	0.6464	0.6660	0.4724	1.0000	0.7834	0.2841	0.7339	-0.0868
count_words_title	0.7996	0.7874	0.7682	0.7936	0.5800	0.7834	1.0000	0.3401	0.7933	-0.0282
count_numbers	0.3641	0.3622	0.3373	0.3512	0.2776	0.2841	0.3401	1.0000	0.3385	-0.1371
count_stopwords	0.9450	0.8974	0.8814	0.9110	0.6350	0.7339	0.7933	0.3385	1.0000	-0.0588
mean_word_len	0.0191	-0.0333	-0.0015	0.0689	-0.0735	-0.0868	-0.0282	-0.1371	-0.0588	1.0000

After text cleaning, we calculated some nlp based features like count_of_words, count_of_letters, etc. However, we did not choose to use them as training features since we would like to make predictions semantically.

Below, you can see the average number of words of label classes.



This is the average title count of labels. Titles possibly are used while indicating important notes or drug names.



For the label encoding phase, we used LabelEncoder() of scikit-learn.

Finally, for train-test-split we chose .75/.25 proportion.

3.3. Text Representation

Let us remind you that in our dataset every row consists of one review and columns(also called as features) size will not be constant, it is depending on your feature creation method. In this study, I tried to implement five different **text representation** methods.

❏ Word Count Vectors:

- ★ **binary=True** → In every cell, there are presence/absence values for each feature.
- ★ **binary=False** → In every cell, there are frequency values for each feature.

❏ Tf-idf Vectors:

It is similar to bow just you need to multiply it with Inverse Document Frequency, that is why these two methods are generally called as “bag of words”.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

With these techniques, every word corresponds to just some scalar value.

Up to here, the order of the words in our reviews are not taken into consideration. Behold! In the following methods, order of the words in a document will not be ignored.

❏ Word Embeddings:

In this method, every word is considered as a vector that’s why we can think of our dataset with 3-dimension. It could be trained in the current dataset or it is possible to use pre-trained word embeddings such as GloVe. I tried both of them.

❏ Text Based Features:

You could create features on your own if you think that it can be helpful to separate categories wisely.

3.4. Model Trainings

For all of the models, we used .75/.25 train-test-split and 5-fold stratified cross validation. For evaluation we prefer to use these metrics:

+ Accuracy: this is the main metric that we used while comparing the models.
+ Precision, recall, f1 score and confusion matrix: these metrics are also evaluated but they are not used for comparison since they are calculated for each label(in our case top-10 labels).

Each of the ML models are trained on three different text representations:

- Occurrence vector: This vector is calculated by using CountVectorizer(binary=True)
- Term frequency vector: This vector is calculated by using CountVectorizer (binary=False)
- tf-idf vector: This vector is calculated by using TfidfVectorizer().

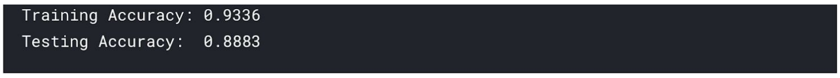
Machine Learning models trained on these vectors are:

1. Naive Bayes
2. Logistic Regression
3. SVM
4. Random Forest
5. Gradient Boosting
6. XGBoost(trained on GPU notebook of Kaggle)

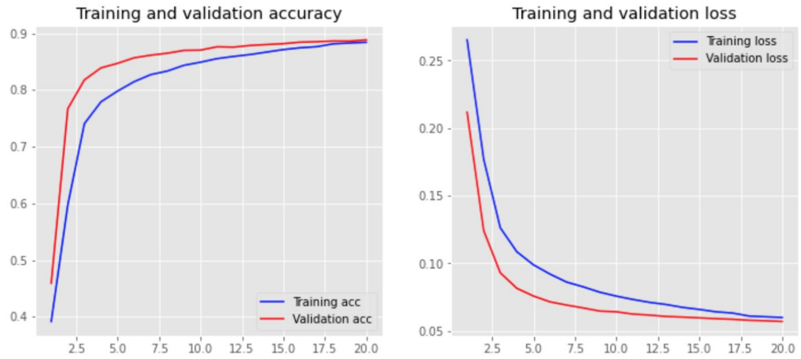
Also for hyperparameter tuning GridSearch() method is used.

We also tried word embeddings using deep learning models. For both of the experiments, we used **relu** as an activation function for all hidden layers. Two different word embeddings techniques are used on GPU notebook of Kaggle:

★ We trained with my own word embedding(dimension=50) using the Embedding layer of Keras. In this experiment we used vanilla neural network GlobalMaxPool layer + DropOut(0.5) layer + Dense(fully connected) layer with 50 nodes + SoftMax(output) layer with 10 nodes.

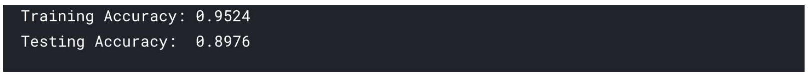


```
Training Accuracy: 0.9336
Testing Accuracy:  0.8883
```

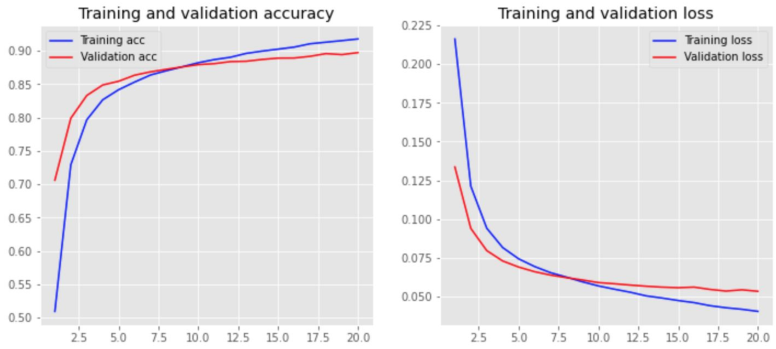


They are converging at some point between 0.88-0.93. It seems that there is no overfitting or underfitting, it is an acceptable model.

- ★ As a second experiment, we used the pre-trained word embedding model Glove(6B 50d) with a CNN model. The architecture of CNN model is Embedding(trainable=True for fine tuning) layer + Conv1D(filters=50, kernel size=3) + MaxPooling1D(3) layer + DropOut(0.5) layer + Dense(fully connected) layer with 50 nodes + SoftMax(output) layer with 10 nodes.



```
Training Accuracy: 0.9524
Testing Accuracy:  0.8976
```

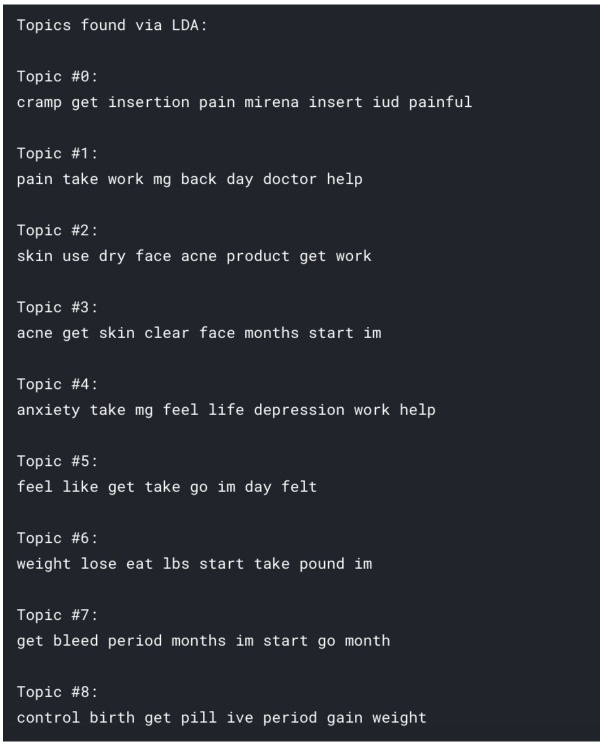


They are converging at some point between 0.95-0.89. It seems that there is no overfitting or underfitting, it is an acceptable model.

3.5. Topic Modelling & Word Cloud & PCA

- LDA

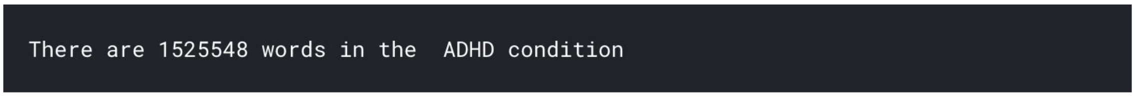
Latent Dirichlet Allocation(LDA) is an example of topic modelling and it is used to classify the text to a particular topic. See topics that are found via LDA. We ran LDA using the tf-idf method.



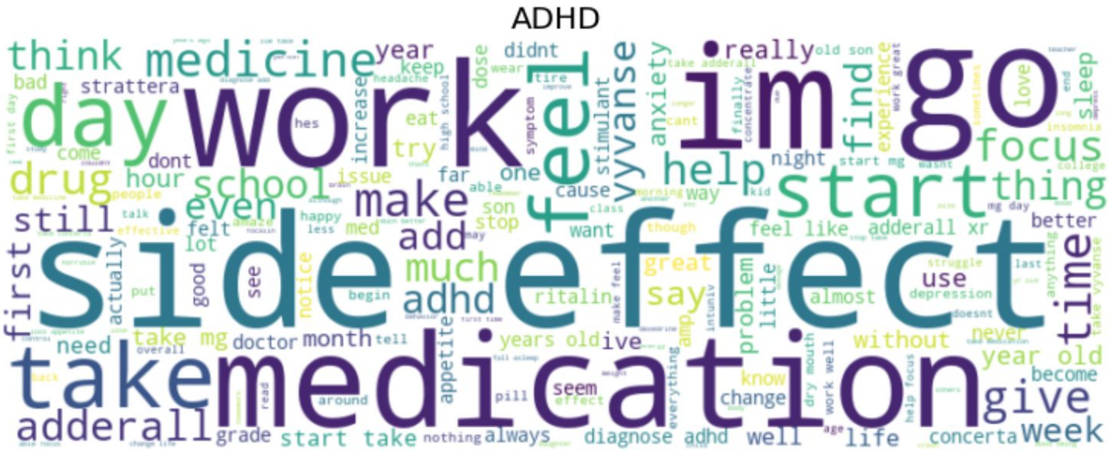
```
Topics found via LDA:

Topic #0:
cramp get insertion pain mirena insert iud painful

Topic #1:
pain take work mg back day doctor help

Topic #2:
skin use dry face acne product get work

Topic #3:
acne get skin clear face months start im

Topic #4:
anxiety take mg feel life depression work help

Topic #5:
feel like get take go im day felt

Topic #6:
weight lose eat lbs start take pound im

Topic #7:
get bleed period months im start go month

Topic #8:
control birth get pill ive period gain weight
```

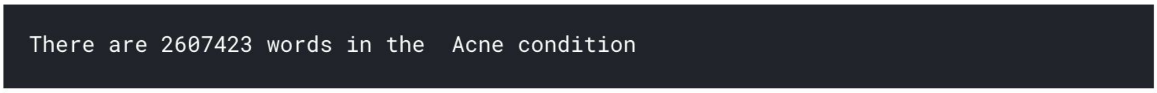
- Word Cloud

You can see the word cloud of all top-10 labels.

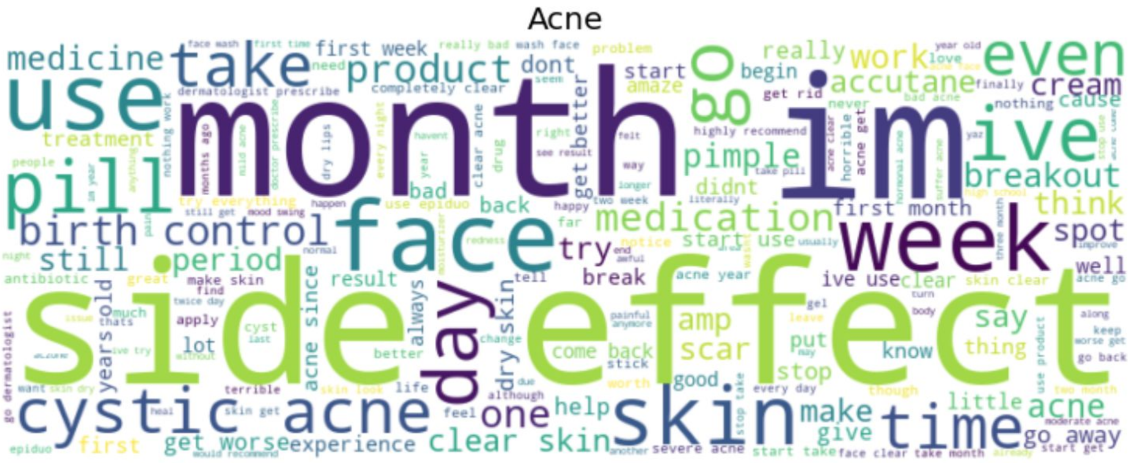


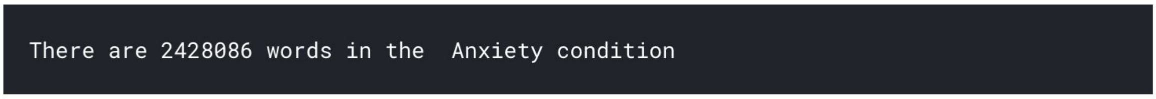
```
There are 1525548 words in the  ADHD condition
```



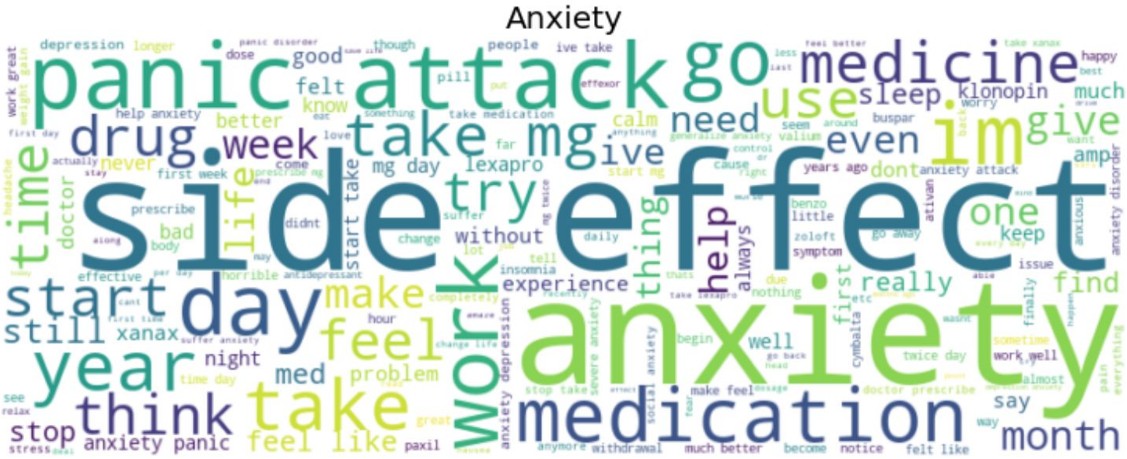


```
There are 2607423 words in the  Acne condition
```

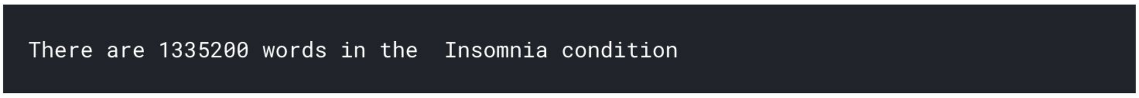




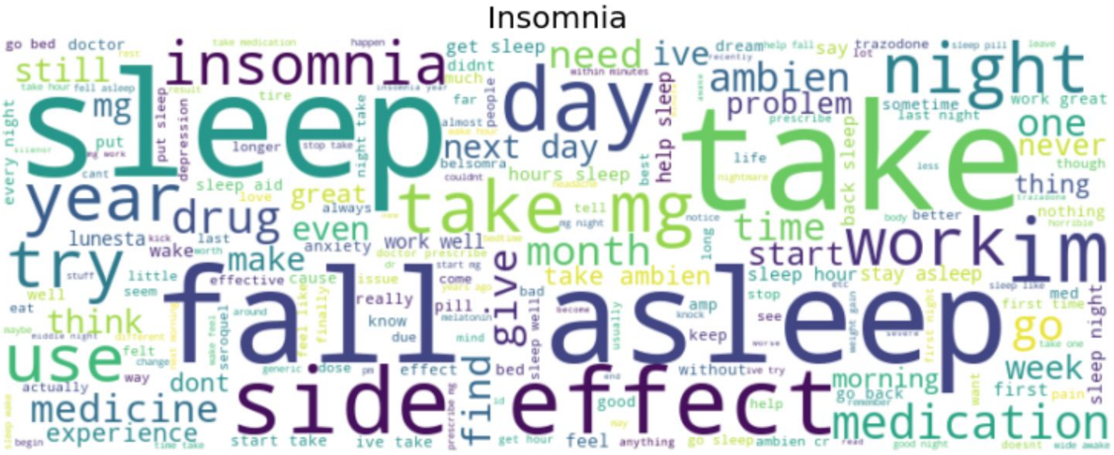
```
There are 2428086 words in the  Anxiety condition
```



Anxiety

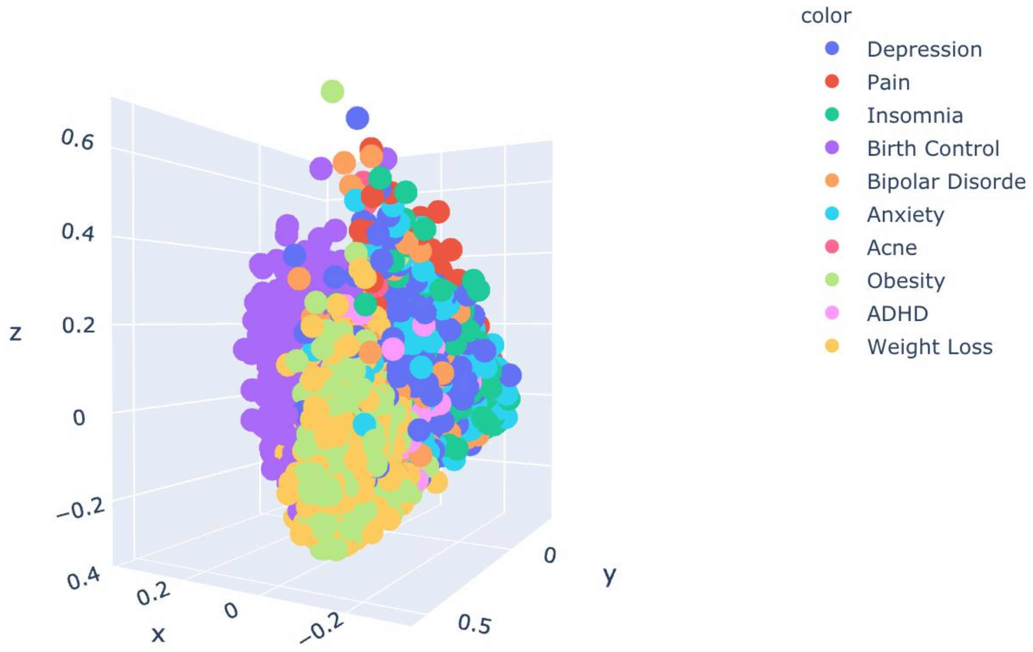


```
There are 1335200 words in the  Insomnia condition
```

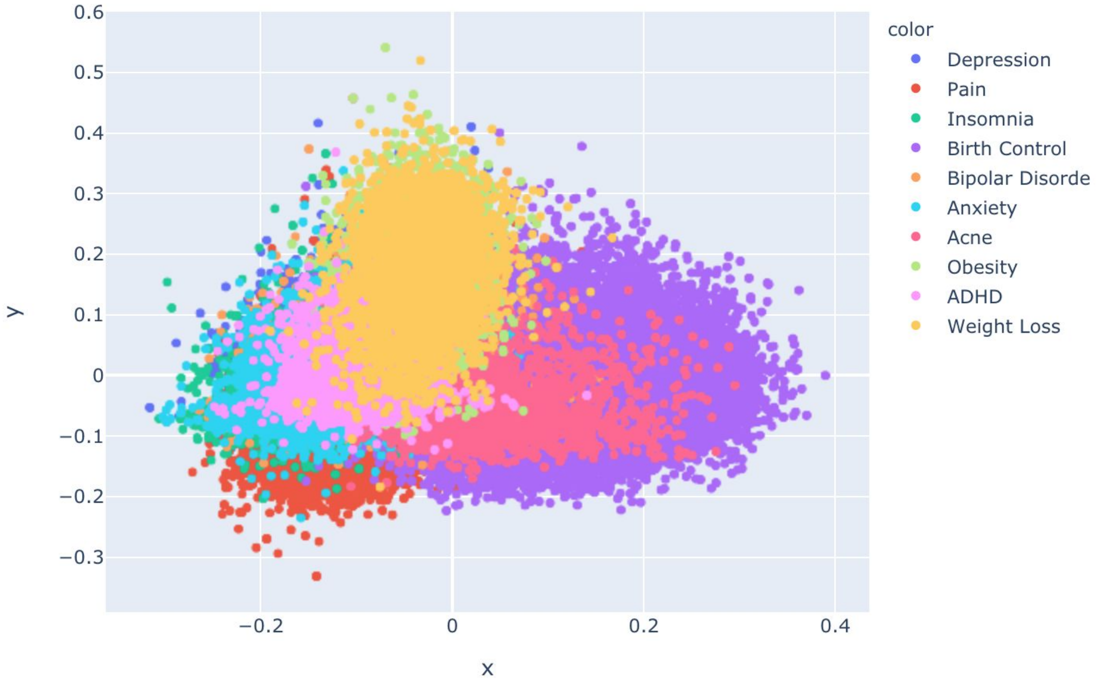


Insomnia

- PCA(3d)


color
Depression
Pain
Insomnia
Birth Control
Bipolar Disorde
Anxiety
Acne
Obesity
ADHD
Weight Loss
z
0.6
0.4
0.2
0
−0.2
x
0.4
0.2
0
−0.2
y
0
0.5

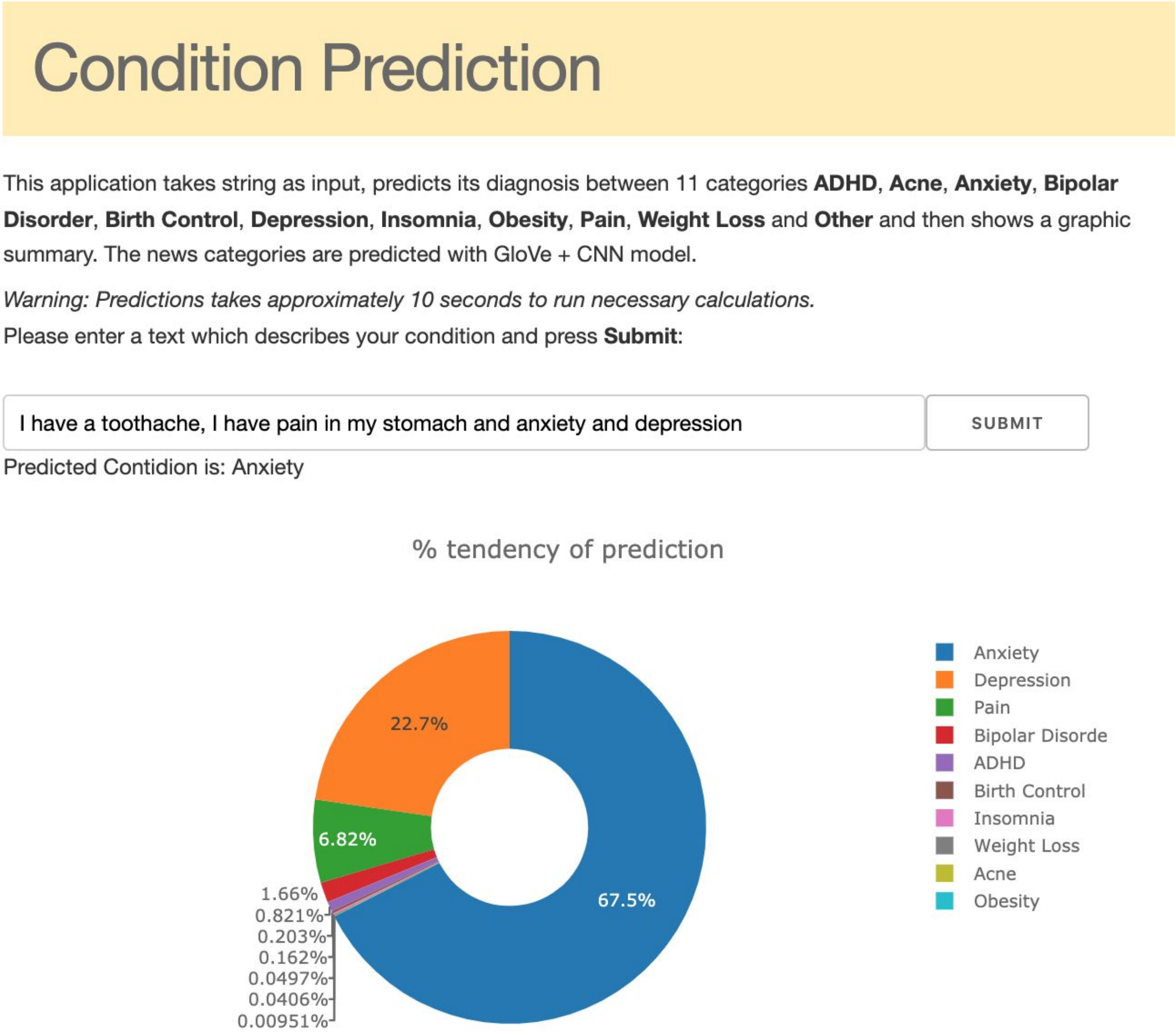
- PCA(2d)


color
Depression
Pain
Insomnia
Birth Control
Bipolar Disorde
Anxiety
Acne
Obesity
ADHD
Weight Loss
y
0.6
0.5
0.4
0.3
0.2
0.1
0
−0.1
−0.2
−0.3
x
−0.2
0
0.2
0.4

3.6. Deployment

In the deployment phase, dash and heroku used to build and deploy the app. Dash is a framework to create single page basic web applications based on flask and plotly. Also, heroku is a free deployment tool.

Please follow the link for app: https://conditionpredictor.herokuapp.com/



Condition Prediction

This application takes string as input, predicts its diagnosis between 11 categories **ADHD**, **Acne**, **Anxiety**, **Bipolar Disorder**, **Birth Control**, **Depression**, **Insomnia**, **Obesity**, **Pain**, **Weight Loss** and **Other** and then shows a graphic summary. The news categories are predicted with GloVe + CNN model.

Warning: Predictions takes approximately 10 seconds to run necessary calculations.

Please enter a text which describes your condition and press **Submit**:

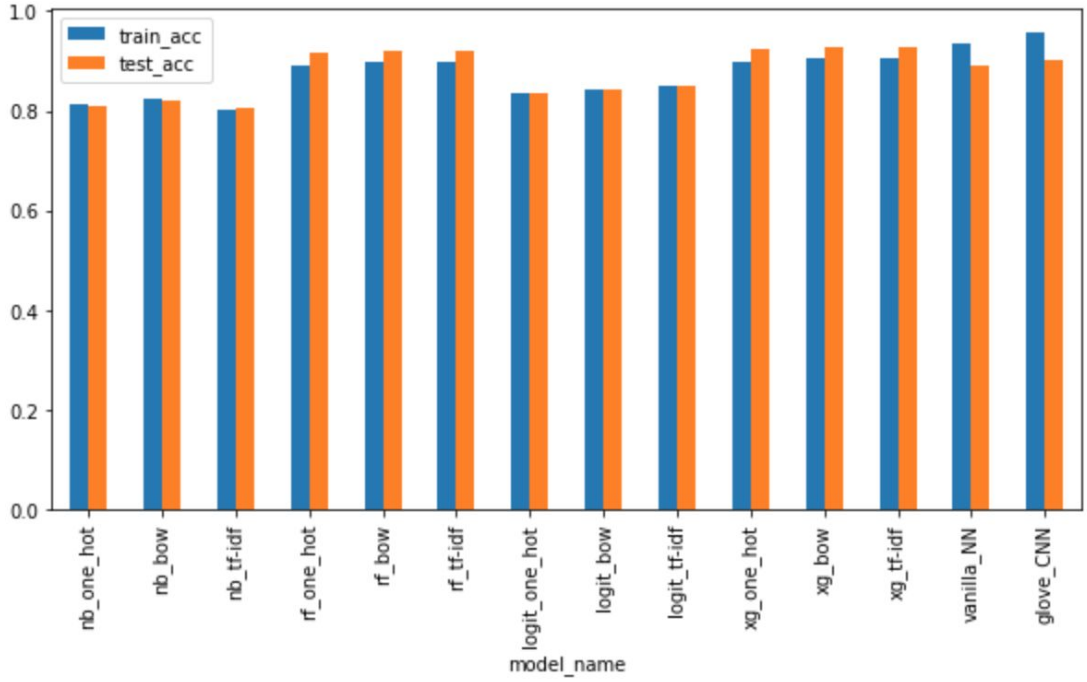
I have a toothache, I have pain in my stomach and anxiety and depression

SUBMIT

Predicted Contidion is: Anxiety

The input text is converted to a pre-trained word embedding model. After that, a CNN classifier is applied to predict its condition.

4. Results



As you see from the above barplot, all models have nice scores due to hyperparameter tuning. Neural networks are the most successful models on train dataset, but on the test set their accuracy scores are a bit off compared to RF and XGBoost. On the other hand training scores of RF and XGBoost are lower than their test scores which is a weird behaviour, so we choose the Glove CNN model over them in our app.

References:

- https://towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf156893c24

- https://realpython.com/python-keras-text-classification/#what-is-a-word-embedding

- https://medium.com/@imamun/creating-a-tf-idf-in-python-e43f05e4d424

- https://towardsdatascience.com/using-deep-learning-for-end-to-end-multiclass-text-classification-39b46aecac81

- https://towardsdatascience.com/text-classification-in-python-dd95d264c802

- https://dash.plotly.com/basic-callbacks