MACHINE LEARNING APPLICATIONS - 2023/24

Bachelor in Data Science and Engineering

DRUGS REVIEWS FINAL PROJECT

ANTHONY MENDOZA - 100529621
DIEGO HERNÁNDEZ SUÁREZ - 100472809
CELIA RAMPÉREZ MARTÍN - 100451825
WOUTER HENDRICKX - 100518303

Instructors:
JERÓNIMO ARENAS GARCÍA
VANESSA GÓMEZ VERDEJO
CARLOS SEVILLA SALCEDO

Date: 7th May, 2024

Contents

1 INTRODUCTION **1**

1.1 Dataset . . . 1

1.2 Objective . . . 1

2 Task 1 : TEXT PREPROCESSING AND VECTORIZATION **2**

2.1 Step 1 : Text Preprocessing . . . 2

2.2 Step 2 : Vectorization . . . 2

2.2.1 BoW vs. TF-IDF representation . . . 3

2.2.2 Word2vec/Glove based representation vs. Doc2Vec vectorization . . . 4

2.2.3 LDA . . . 4

3 Task 2 : MACHINE LEARNING MODEL **6**

3.1 Classification . . . 6

3.2 Regression . . . 8

4 DASHBOARD **11**

5 CONCLUSIONS **12**

6 REFERENCES **13**

1 INTRODUCTION

1.1 Dataset

For this project, after an intense search, and hesitating between several datasets, we finally decided to use the dataset on drugs and their beneficial and side effects. Specifically, the dataset provides patient reviews on specific drugs along with related conditions. Reviews are grouped into reports on the three aspects: benefits, side effects and general comments. In addition, ratings are available on overall satisfaction, as well as a 5-step side effect rating and a 5-step effectiveness rating. The data was obtained by crawling online pharmaceutical review sites, although we obtained the whole set directly from the UC Irvine University Machine Learning Repository 1. The set consists of 8 variables and 4143 instances, divided into 3107 (75%) for training, and 1036 (25%) for testing. Regarding the 8 variables, we have:

- urlDrugName (categorical): name of drug
- rating (numerical): From 0 to 10-star patient rating
- effectiveness (categorical): 5-step effectiveness rating
- sideEffects (categorical): 5-step side effect rating
- condition (categorical): name of condition
- benefitsReview (text): patient on benefits
- sideEffectsReview (text): patient on side effects
- commentsReview (text): overall patient comment

Another important thing and that we took very much into account when choosing this data, is that the dataset does not contain missing values, which we consider very beneficial for the future work we will perform on it, since we will not take the risk of introducing biases to the data that may worsen the quality of our analysis or cause the application of our machine learning models to be less effective.

1.2 Objective

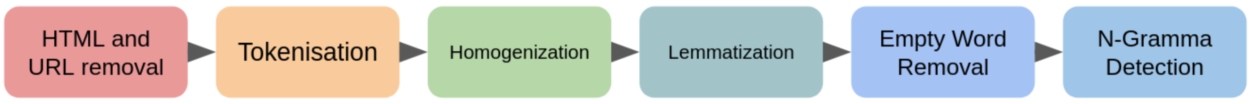
Our project's primary goal is to apply advanced techniques in natural language processing (NLP) and machine learning (ML) to analyze and extract valuable information from a dataset that includes reviews about medications. By utilizing this data, we hope to better understand the relationship between patients' opinions about the medications and the quantitative evaluations of effectiveness and side effects. However, we will also employ other variables in order to obtain more reliable and realistic results in order to understand the impact of each drug for the different patients.

With this dataset, we will use several processing and models in orders to make predictions. First, the aim will be to preprocess the reviews in order to make them usable with our models. We will discuss more deeply this step in Task 1. Then, we will do a classification and a regression with one or many of the text variables in order to be able to predict from these text data the numerical features such as the rating. Furthermore, we will employ regression techniques in order to predict differently the expected rating that a patient will assign to a specific drug, after that we will compare both methods in order to see which one is more recommended applying.

2 Task 1 : TEXT PREPROCESSING AND VECTORIZATION

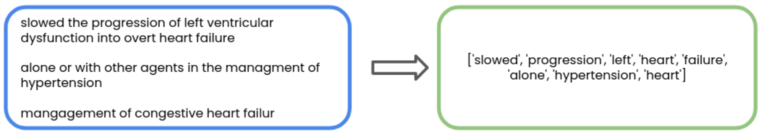
2.1 Step 1 : Text Preprocessing

To start working with our data, we needed to deal with the text variables first. The text preprocessing pipeline implemented aims to prepare the data for efficient and accurate analysis. This process is crucial to reduce the noise and variability of the natural text, facilitating the extraction of relevant features for further vectorization. The steps followed in our preprocessing pipeline are detailed below :



Each of these steps helps transform the raw text into a more structured and analyzable form, optimizing both the accuracy and efficiency of subsequent thematic and vector analysis.

At the end of the preprocessing, we get for example this kind of transformation :


slowed the progression of left ventricular dysfunction into overt heart failure
alone or with other agents in the managment of hypertension
mangagement of congestive heart failur
['slowed', 'progression', 'left', 'heart', 'failure', 'alone', 'hypertension', 'heart']

2.2 Step 2 : Vectorization

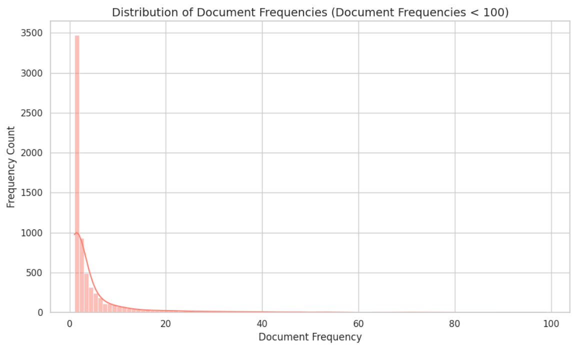
The next step for processing our data is to vectorize them in order to make them usable with our models. This step is crucial since it is the key features that allow machine learning to deal with text data. We will use several methods so that we will be able to compare them with prediction models to see which method is better. Of course, we know already that certain methods provide better result, so we would be able to retrieve these conclusions or not with our proper dataset. We will compare BoW/TF-IDF with Word2vec/Glove and with LDA.

Before apply these methods, we will clean our dataset. Indeed, some reviews must have specific words inside and on the other side, some words might be appearing in every document. Since we want to apply topic modeling, we need to keep only the word that allow us to identify a specific group of reviews.

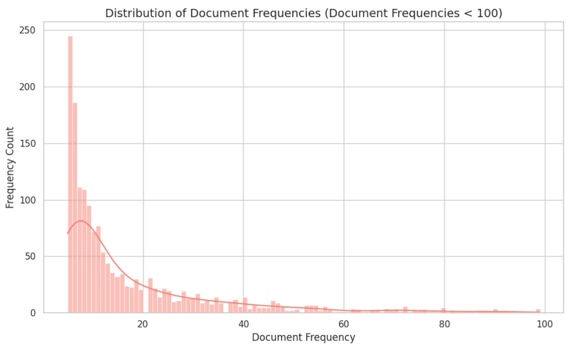
To do that, we will create a range that remove all extreme word frequency, thus trying to avoid underfitting and overfitting. The method that we have chosen was to remove the word present in less than 5 reviews and in more than 80%. We used this values because 5 is a standard values used in the literature that provide enough performances and 80% is in the range of 50% and 90% which is also a range widely used in the literature. We specifically have chosen 80% because since there are no many words per reviews, decreasing to 50% would have removed too many words. Then, we obtain this new distribution of words in the reviews.

- word that appear in less than 5 reviews
- word that appear in more than 80% of the reviews

By doing this cleaning, we decrease significantly the amount of word in our dictionary. We go from 6952 tokens to 1735.



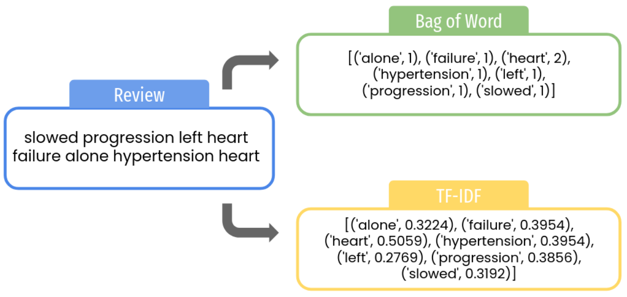
(a) Word frequency



(b) Word frequency after cleaning

2.2.1 BoW vs. TF-IDF representation

We will first use the BoW and TF-IDF representation. To apply these methods, we will use the libraries NLTK and Gensim. First, to create either BoW or TF-IDF, we need to gather all the words of each text features in order to create the vocabulary. Indeed, if we omit some word, the representation will no longer be the same since the representation of a word with BoW or TF-IDF needs to be the length of the size of the vocabulary. Once we have done the representation, we can use them for Word2Vec or LDA for example. However, once think we need to keep in mind is that these representations need to be charging as sparse matrix, since the whole matrix can contain millions of elements.

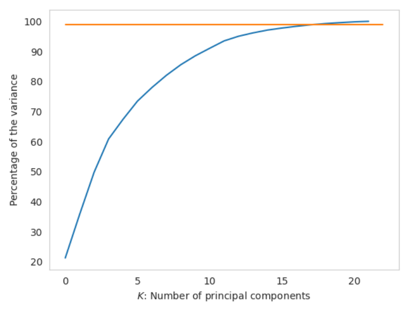

Review
slowed progression left heart failure alone hypertension heart
Bag of Word
[('alone', 1), ('failure', 1), ('heart', 2), ('hypertension', 1), ('left', 1), ('progression', 1), ('slowed', 1)]
TF-IDF
[('alone', 0.3224), ('failure', 0.3954), ('heart', 0.5059), ('hypertension', 0.3954), ('left', 0.2769), ('progression', 0.3856), ('slowed', 0.3192)]

2.2.2 Word2vec/Glove based representation vs. Doc2Vec vectorization

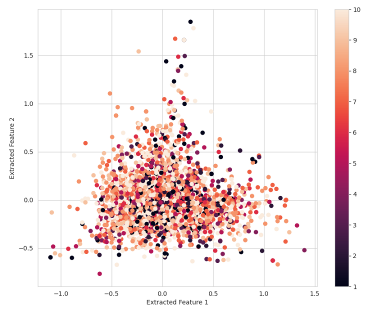
Word2Vec. To obtain the embeddings, in particular with Word2vec, we will proceed by features extraction in order to get the best embedding possible. To initialize a Word2Vec, we need to configure 3 parameters :

- **'sg'** : which specifies the training algorithm: CBOW (Continuous Bag of Words) when sg=0 and Skip-Gram when sg=1. We will use sg=1 since it's the best approach according to the literature.
- **'size'**, which indicates the size of the embedding vectors, i.e. the dimensionality of the feature vectors. For this parameter, we tried with an embedding of size 100 and analyze it using features extraction.
- **'window'**, which indicates the size of the context window that determines the maximum distance between the current word and the predicted word within a sentence. For this parameter we set it up to 16 since, after cross validation, we see that there is no clear improvement.

Then, in order to use the best size of the embedding, we will use feature extraction such as PCA. After setting the PCA, we get that 99% of the variance of the embedding is explained by only 18 components. So we will use, for Word2Vec, an embedding size of 18. We can also plot the embeddings in the 2D plan and color each document according to its associated rating. The obtained plot shows us that all the rating are really packed together. So it might be difficult to well predict this variable with Word2Vec.



(a) Word frequency



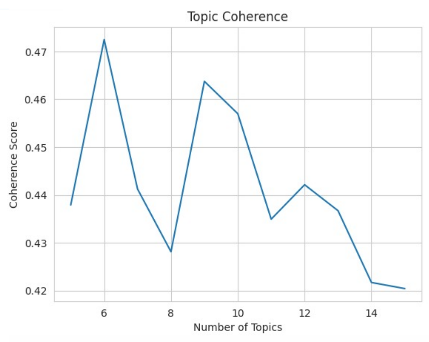
(b) Word frequency after cleaning

Doc2Vec. Similarly, we created the embeddings with Doc2Vec. In this case, we employed the TaggedDocument function to create a list of documents. Subsequently, we had to determine the ideal vector size and the number of epochs for our data to find the best Doc2Vec model. We trained different configurations of the model, iterating over various epochs and vector sizes, and then evaluated the performance. Our best model for this embedding resulted in 50 number of epochs with a vector size of 200.

2.2.3 LDA

In this study, we used the Gensim library's Latent Dirichlet Allocation (LDA) algorithm to extract themes and represent documents as vectors, starting with tokenizing the text corpus and using the previously generated BoW representation as input for our LDA.

After creating the BoW corpus, we trained the LDA model, adjusting parameters such as the number of topics and passes through the corpus to optimize theme extraction. We used topic coherence to determine the optimal number of topics, evaluating a range from 5 to 15 to balance depth of analysis with computational efficiency. Our goal was to select the best number of topics for a robust and interpretable model without significant delays, ensuring a methodical approach.

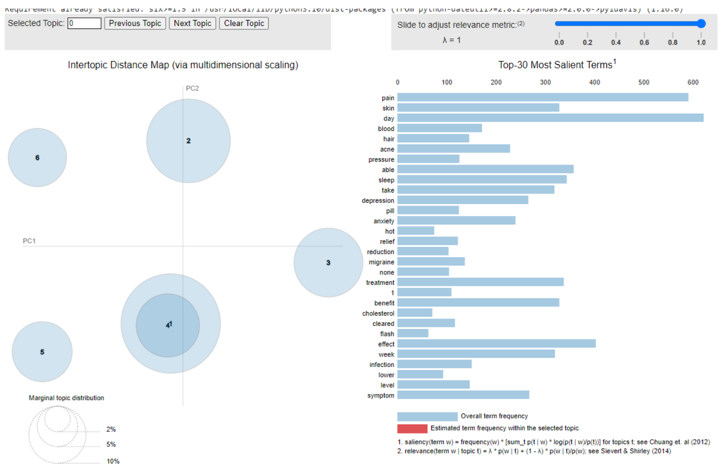


As we can see, the number of topics with the highest score of coherence is at 6 topics, so that will be the value that we will use in order to develop our models employing LDA.

For visualization and analysis, we plotted the vector representations of documents to show each one's thematic composition. These plots highlight the significance of each theme within individual documents, aiding in understanding the corpus's thematic structure.

Vector Representation of Document 3: **Topic 0: 0.4125 Topic 1: 0.3704 Topic 2: 0.1969**

In our LDA analysis of Document 3, the thematic composition is distributed across three main topics. Topic 0 is the most dominant, making up about 41.25% of the document's content, suggesting high relevance to the document's context. Topic 1 follows with 37.04%, also significantly shaping the document's themes. Topic 2, though the least dominant, still contributes 19.69%. This distribution offers a detailed view of the document's thematic structure, highlighting the significant interplay of topics that define its content. Notably, Topics 3, 4, and 5 are absent from the vector representation of Document 3, indicating their minimal impact on this document. Within the LDA model's broader scope of six topics, only those with substantial relevance and higher probability scores appear in the document's vector representation.



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

The interactive graph demonstrates how the Latent Dirichlet Allocation (LDA) model organizes topics based on the specified number of topics. Each topic includes tokens with specific weights that reflect their importance. For example, a notable topic associated with dermatological conditions features prominent keywords like "acne" and "skin." The graph allows for parameter adjustments such as the relevance metric, showing the 30 most salient terms for a topic in the bar plot when hovered over. This graphical representation provides clear insights into how certain tokens are connected to specific topics, facilitating an easier and more intuitive understanding of the model's thematic structure.

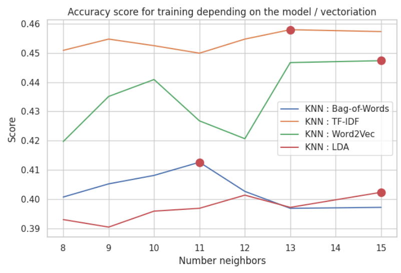
3 Task 2 : MACHINE LEARNING MODEL

3.1 Classification

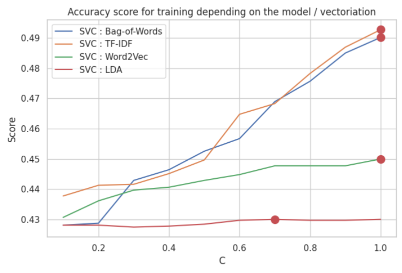
The first algorithm we will try is to classify the reviews per *effectiveness*. We will try to see which type of review is the best to predict the effectiveness of a drug. We thought that it could be a real problem in the case of a new developed drug, where we only have reviews of the testing part. So, in order to achieve this task, we will proceed by cross validation to get the best model for each type of review : *benefitsReview*, *sideEffectsReview* and *commentsReview*. To make a classification, we have multiple type of model. For our purpose, we will try a KNN and an SVC. The first one will be to see if a simple model can reach good performances, and the second one is to check whether a more complex one can increase the performances. To find the best model and the best review, we will use a cross validation method. First, we will test with large values for parameters. In fact, for the SVC, we cross validate the parameters C by using logarithm scale between 10^0 and 10^5. For the KNN, we will cross validate the number of neighbors *n_neighbors* between 1 and 10. Then, we will take C between 0.1 and 1 for the SVC and search around 10 neighbors for KNN, since the accuracy seams to increase in this range.

We get as first results these plots for the feature *benefitsReview* :

So as we can see, as a first look, the best accuracy is obtained with SVC and Bag of Words. This could be because of the efficiency in linearly separable cases. Indeed, SVC works well when there is a clear boundary between features, which is exactly the case with BoW.

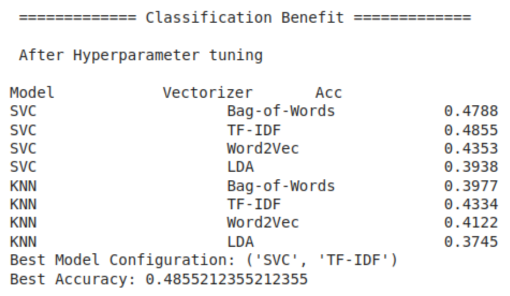


(a) Accuracy for KNN for benefit



(b) Accuracy for SCV for benefit

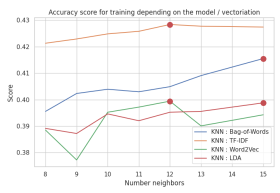
And this accuracy with the best model for each mode architecture and vectorization :

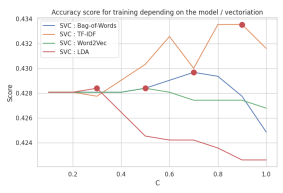

```
============= Classification Benefit =============

 After Hyperparameter tuning

Model              Vectorizer       Acc
SVC                        Bag-of-Words             0.4788
SVC                        TF-IDF                   0.4855
SVC                        Word2Vec                 0.4353
SVC                        LDA                      0.3938
KNN                        Bag-of-Words             0.3977
KNN                        TF-IDF                   0.4334
KNN                        Word2Vec                 0.4122
KNN                        LDA                      0.3745
Best Model Configuration: ('SVC', 'TF-IDF')
Best Accuracy: 0.4855212355212355
```

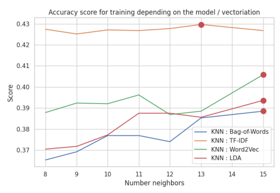
So as we can see, the accuracy is quite low. We could explain it by multiple facts. First, BoW seems to give the best results, however, it is terrible to take into account the frequency of words among other texts and the context of a word. So, with this technic, we can't get good results in NLP. We can also see that the LDA doesn't give good performances. In fact, it is the worst model so far. This can be explained by the fact that our reviews don't have many words inside, and thus is it quite difficult to deduce topics from them. Moreover, because they are all related to drugs and thus don't have many differences between each others, the topic modeling can't find clear bounds between them.

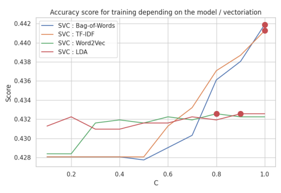
However, we can compare the performances with the other features such as the *sideEffectsReview* and *commentsReview* :

We get these plots of the accuracy :


(a) KNN for side

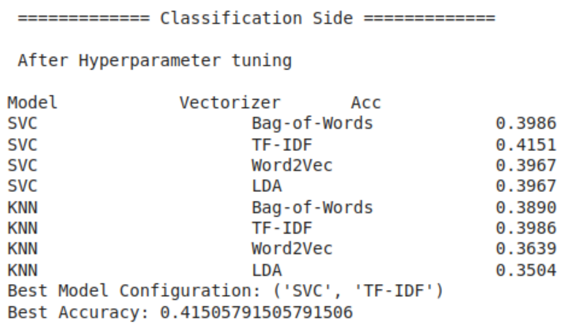

(b) SVC for side


(c) KNN for comment


(d) CV for comment

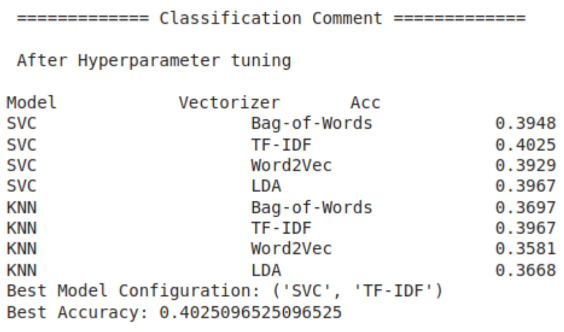
And these performances with the best model :

So after applying these different vectorization technics, we can see that the best features to predict the rating is the *sideEffectsReview* with an SVC and TF-IDF. It makes sense since SVC is a more sophisticated model than



```
============= Classification Side =============

 After Hyperparameter tuning

Model            Vectorizer       Acc
SVC                     Bag-of-Words               0.3986
SVC                     TF-IDF                     0.4151
SVC                     Word2Vec                   0.3967
SVC                     LDA                        0.3967
KNN                     Bag-of-Words               0.3890
KNN                     TF-IDF                     0.3986
KNN                     Word2Vec                   0.3639
KNN                     LDA                        0.3504
Best Model Configuration: ('SVC', 'TF-IDF')
Best Accuracy: 0.41505791505791506
```

(a) Accuracy for side

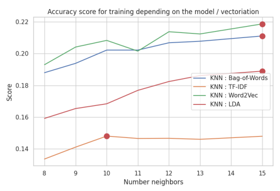


```
============= Classification Comment =============

 After Hyperparameter tuning

Model            Vectorizer       Acc
SVC                     Bag-of-Words               0.3948
SVC                     TF-IDF                     0.4025
SVC                     Word2Vec                   0.3929
SVC                     LDA                        0.3967
KNN                     Bag-of-Words               0.3697
KNN                     TF-IDF                     0.3967
KNN                     Word2Vec                   0.3581
KNN                     LDA                        0.3668
Best Model Configuration: ('SVC', 'TF-IDF')
Best Accuracy: 0.4025096525096525
```

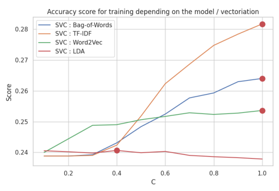
(b) Accuracy for comment

KNN and the embedding of Word2Vec / LDA are more useful with NN. Moreover, the best performances are obtained with TF-IDF since it is more powerful than BoW by its construction and SVM are originally designed to work with sparse feature representations, like TF-IDF and less with dense features like Word2Vec or LDA. If we look at the other variables, we can see that they give lesser results than benefitsReview. We can maybe explain that because in general, there are less information in these reviews and that the effectiveness of a drug depends more on its benefits than on the comment or side effects. However, as we can see, the predictions are quite low. So we will check if we can improve the performances by combining several features.

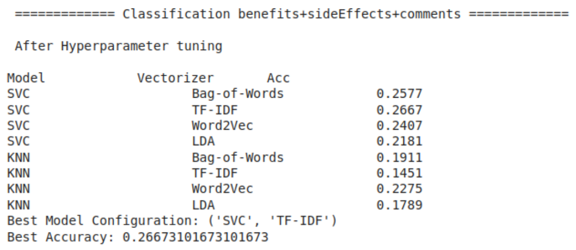
We will take a look at the combination of the *benefitsReview*, *sideEffectsReview* and *commentsReview* in order to see if combining multiple features increases the results. We get, after cross validation, these results :



(a) KNN for side



(b) KNN for comment



```
============= Classification benefits+sideEffects+comments =============

 After Hyperparameter tuning

Model            Vectorizer       Acc
SVC                     Bag-of-Words               0.2577
SVC                     TF-IDF                     0.2667
SVC                     Word2Vec                   0.2407
SVC                     LDA                        0.2181
KNN                     Bag-of-Words               0.1911
KNN                     TF-IDF                     0.1451
KNN                     Word2Vec                   0.2275
KNN                     LDA                        0.1789
Best Model Configuration: ('SVC', 'TF-IDF')
Best Accuracy: 0.26673101673101673
```

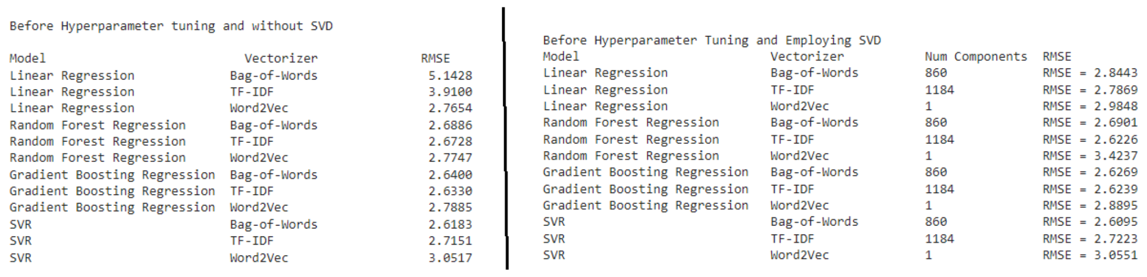
(c) Accuracy for benefit + review + comment

So as we can see, the doesn't increase even by merging all the features together. We can explain that by the fact that the comment and the benefit don't reach good performances on their own. So by adding to the side effects, it doesn't improve the results.

3.2 Regression

We developed multiple models using different vectorizers—Bag-of-Words, TF-IDF, and Word2Vec—to evaluate their impact on a regression task aimed at predicting the "rating" variable, which scores drugs from 1 to 10 based on key review terms. Our selection included Linear Regression, Random Forest, Gradient Boosting, and Support Vector Regression, chosen to explore diverse scenarios and effectiveness measured by RMSE. We also investigated the impact of dimensionality reduction techniques such as Singular Value Decomposition (SVD) and implemented validation methodologies to optimize and enhance model performance.

For this first try, we only used *benefitsReview* as the prediction variable. But for the cross validation, we aspire to try with different columns in order to predict the target variable *rating*. In this case, we obtained the following



```
Before Hyperparameter tuning and without SVD

Model                          Vectorizer             RMSE
Linear Regression             Bag-of-Words            5.1428
Linear Regression             TF-IDF                  3.9100
Linear Regression             Word2Vec                2.7654
Random Forest Regression      Bag-of-Words            2.6886
Random Forest Regression      TF-IDF                  2.6728
Random Forest Regression      Word2Vec                2.7747
Gradient Boosting Regression  Bag-of-Words            2.6400
Gradient Boosting Regression  TF-IDF                  2.6330
Gradient Boosting Regression  Word2Vec                2.7885
SVR                           Bag-of-Words            2.6183
SVR                           TF-IDF                  2.7151
SVR                           Word2Vec                3.0517
```

```
Before Hyperparameter Tuning and Employing SVD
Model                         Vectorizer          Num Components  RMSE
Linear Regression             Bag-of-Words        860             RMSE = 2.8443
Linear Regression             TF-IDF              1184            RMSE = 2.7869
Linear Regression             Word2Vec            1               RMSE = 2.9848
Random Forest Regression      Bag-of-Words        860             RMSE = 2.6901
Random Forest Regression      TF-IDF              1184            RMSE = 2.6226
Random Forest Regression      Word2Vec            1               RMSE = 3.4237
Gradient Boosting Regression  Bag-of-Words        860             RMSE = 2.6269
Gradient Boosting Regression  TF-IDF              1184            RMSE = 2.6239
Gradient Boosting Regression  Word2Vec            1               RMSE = 2.8895
SVR                           Bag-of-Words        860             RMSE = 2.6095
SVR                           TF-IDF              1184            RMSE = 2.7223
SVR                           Word2Vec            1               RMSE = 3.0551
```

Figure 7: RMSE for Benefits Reviews: Comparison of Results Without and With SVD Applied, Respectively

results:

From the presented data, it is evident that the combination of the Support Vector Regressor (SVR) with the Bag-of-Words (BoW) vectorization technique outperformed other model-vectorizer pairs. We get a Root Mean Square Error (RMSE) of 2.6183. This result, observed in the initial evaluation phase prior to the application of any robust validation methodologies, is promising and suggests a good baseline performance of the SVR with BoW.

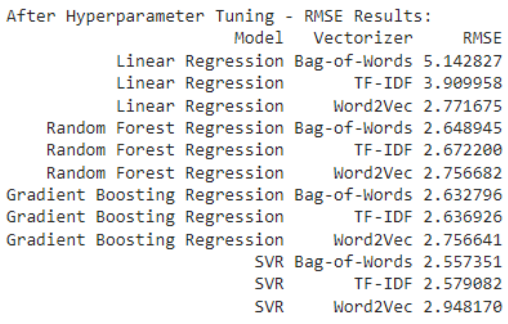
However, we decided to employ SVD algorithms in order to compare the results and decide if it was worth it to include dimension reduction techniques into the models. For that reason, we obtained the optimal number of components for each of the vectorizers. Then, we created a scree plot in order to see which was the most recommended value for the number of components, in our case it were BoW (860), TF-IDF (1184) and Word2Vec (1), it is possible to see in the script those graphs.

Models using Bag-of-Words and TF-IDF vectorizations benefit from SVD, as shown by significant RMSE reductions in Linear Regression models (e.g., BoW RMSE decreased from 5.1428 to 2.8443). Although the improvements are modest for some models, such as the SVR with BoW where RMSE improved slightly from 2.6183 to 2.6095, these are still positive changes. However, SVD does not consistently enhance performance for Word2Vec; in some instances, it even degrades it. This indicates that SVD might remove important information or disrupt key data structures essential for models like Random Forest or SVR, demonstrating that SVD's effectiveness varies based on the vectorizer used.

Dense vectorizers like Word2Vec produce rich outputs that can be degraded by SVD, which might remove vital information. In contrast, sparse vectorizers like Bag-of-Words and TF-IDF benefit from SVD, which simplifies high-dimensional, sparse matrices by isolating key features and reducing noise.

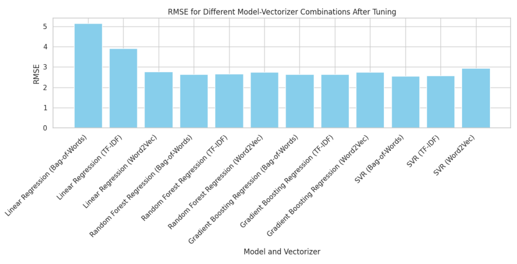
To improve outcomes, we plan to conduct further experiments using varied textual features from different dataset columns and implementing advanced hyperparameter tuning. These efforts aim to optimize model configurations for better predictive accuracy.

Our initial aim was to integrate cross-validation with SVD for suitable vectorizers. Unfortunately, incorporating truncated SVD into our pipeline extended processing times to over 10 hours and often caused execution timeouts. This outcome was frustrating, especially after considerable effort to enhance our models with this technique. While SVD offered some benefits for models with high RMSE, it adversely affected those using the Word2Vec vectorizer. Given these mixed results, we decided that excluding SVD from our pipeline was a practical choice,



```
After Hyperparameter Tuning - RMSE Results:
                        Model    Vectorizer      RMSE
            Linear Regression Bag-of-Words 5.142827
            Linear Regression       TF-IDF 3.909958
            Linear Regression     Word2Vec 2.771675
     Random Forest Regression Bag-of-Words 2.648945
     Random Forest Regression       TF-IDF 2.672200
     Random Forest Regression     Word2Vec 2.756682
 Gradient Boosting Regression Bag-of-Words 2.632796
 Gradient Boosting Regression       TF-IDF 2.636926
 Gradient Boosting Regression     Word2Vec 2.756641
                          SVR Bag-of-Words 2.557351
                          SVR       TF-IDF 2.579082
                          SVR     Word2Vec 2.948170
```

(a) RMSE for Benefits Review

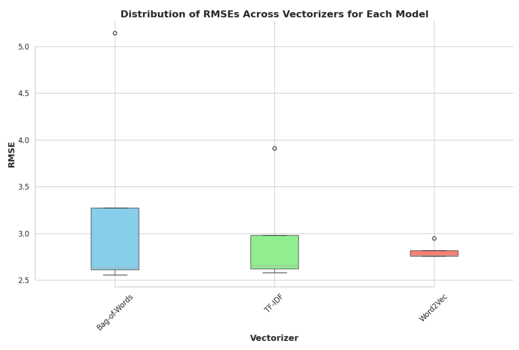


(b) Bar plot Comparison of RMSE Across Different Model-Vectorizer Combinations

as it did not significantly compromise the final outcomes.

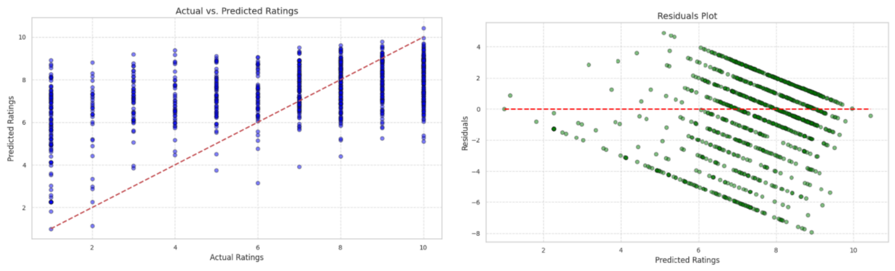
For this regression task, we limited our predictor to the "Benefits Reviews" column due to the high computational demands, as training took over three hours for just this variable. This restriction was crucial to manage training within our resource limits, allowing us to utilize various models and validation methods. Despite these constraints, we were able to effectively plot and analyze the outputs.

From the results, we could extract different insights. The most relevant one was to see that the best model-vectorizer duo was the model SVC with the vectorizer Bag-of-Words, which obtained a RMSE of 2.557351. However, we can notice that the best regressors were firstly SVR and after him, Gradient Boosting. In the third position and close to the second one it will be placed Random Forest Regression and in the last place, Linear Regression. We could also show as a box plot the RMSE for each of the vectorizers in order to see the impact of each one through the different models:

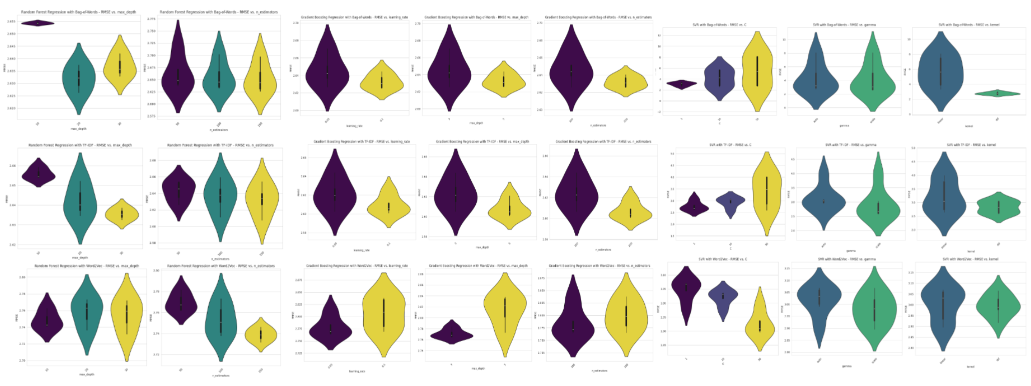


The distributional visualization of RMSEs across different vectorizers reveals key insights into the performance consistency of text representation methods in machine learning models. Notably, the Word2Vec vectorizer shows almost uniform RMSE across all models, indicating consistent performance regardless of the model used. This suggests that Word2Vec provides a stable representation of text data, without significant performance variations due to different underlying algorithms. In contrast, TF-IDF and Bag-of-Words (BoW) vectorizers display greater fluctuations in RMSE, with BoW exhibiting the widest range. This highlights that while BoW can achieve very low RMSEs with certain model setups, making it highly effective, it can also yield poor results if the model configuration is not well-suited to this type of vectorization.

We will illustrate how the best model forecasted ratings and residuals between predicted value and actual observations.



The analysis of the plotted data reveals interesting patterns, showing a weak correlation between predicted and actual ratings, though there are some similarities in their trends. This indicates that while the model captures key underlying features affecting the outcomes, it struggles with accurate predictions due to significant variability around actual results. The discrepancy between actual and predicted ratings highlights challenges in both accuracy (spread along the line of fit) and precision (closeness of data points), pointing out the high variance in our predictions and its impact on model reliability. It is important to mention that we decided to do not apply regression to other columns such as *sideEffectsReview* or *commentsReview* due to the huge amount of time it took to compute the training tasks. The training took more than 3 hours only for the *benefitsReview*.



In the graph, we can see the impact of the different hyperparameters on the RMSE in our regression task. Given that the hyperparameters involved were discrete variables, we chose to utilize violin plots for this analysis. These plots effectively illustrate how the RMSE varies with different values of the hyperparameters. As depicted in the graphs, it is evident that the choice and specific values of the hyperparameters significantly influence the increase or decrease in RMSE. We can clearly see the impact of choosing a certain max depth in models like Random Forest Regression, where the RMSE increases significantly if we choose the max depth with the lowest value.

4 DASHBOARD

We implemented a dashboard to visualize various aspects of our Drugs Reviews dataset. The dashboard features main tabs: LDA topic visualization, topic-document similarity heatmap, and classification model evaluation for SVC and KNN.

- Tab 1: In this tab we integrated an interactive visualization of the topics identified by the LDA model applied to the dataset corpus. This representation consists of a bar chart indicating the number of documents associated with each predominant topic, providing us with a quick view of the distribution of documents across topics and a pyLDAvis graphical representation of the whole LDA model, showing the topics and their distribution in the corpus. The pyLDAvis representation changes based on which topic is selected, by clicking on a bar in the bar chart.
- Tab 2: In this second tab, we implemented a heat map representing a similarity matrix between documents and topics generated by the LDA model. Each cell of the matrix indicates the probability that a document belongs to a particular topic, allowing us to provide the user with an intuitive visualization of how documents are distributed across topics.
- Tab 3: In this third tab of the dashboard we implemented the evaluation of two types of classification models, SVC (Support Vector Classifier) and KNN (K-Nearest Neighbors), applied to different dataset configurations. This tab includes a drop-down menu in which users can select the specific dataset (benefits, side effects, etc.) to visualize the classification results. After selecting the model and the dataset, the tab generates graphs showing the performance (accuracy score) of different model configurations, such as variations in the C parameter for SVC and the number of neighbors for KNN, as well as highlighting the maximum value achieved to facilitate the identification of the best configuration.
- Tab 4: In the final tab of the dashboard we show the influence of hyper-parameter tuning on the RMSE for Random Forest Regression, Gradient Boosting Regression, SVR. Each regression type is displayed with the different vectorizers and the tuned parameter.

This dashboard is a comprehensive tool for analyzing and presenting our results, enabling visual interpretation of models and comparative evaluation of classification strategies. It supports real-time data interaction and customizable visualization parameters, making it efficient for data exploration and presentation.

5 CONCLUSIONS

With this project, we pointed out the significance of opting for an appropriate vectorizer and model pairing according to the specifications of our data. Indeed, depending on the task we want to do and the model we want to use, some vectorizers will be more appropriate than other ones. However, the training part still requires delicate hyperparameter tuning in order to reach good performances. Moreover, we pointed out the importance of cleaning and reducing our data in order to increase performances and reduce time consumption. We have also shown that textual data with not many samples or samples with few words inside are difficult to use in NLP tasks because we can't reach sufficient performances for tasks like classification or regression.
Conversely, Word2Vec and LDA can be a more reliable choice for applications that seek consistent outcomes without extensive tuning efforts across different models. Thus, they could have been more powerful than BoW or TF-IDF, but they are more convenient with neural network.
Finally, as a continuation of this project, we can consider using neural network in order to increase our performances and use the whole potential of sophisticated vectorizers, in this way we could consider more approaches in order to aim for the best models.

6 REFERENCES

1. Kallumadi,Surya and Grer,Felix. (2018). Drug Reviews (Druglib.com). Retrieved 12/04/2024. UCI Machine Learning Repository. Obtained from: https://doi.org/10.24432/C55G6J

2. Vanam, L. (2020). Price Prediction NLP Regression (GitHub). Retrieved 01/05/2024. Obtained from: https://github.com/LaxmiVanam/PricePrediction-NLP-Regression

3. Basic threshold for frequency of stop word (present in less than 5 reviews and in more than 80% of the reviews)

4. NLP Text Classification using SVM

5. Vijay, I. (2021). NLP Text Classification Model (GitHub). Retrieved 01/04/2024. Obtained from: https://github.com/vijayaiitk/NLP-text-classification-model