**Project 1**

Wei Zhou 505650843

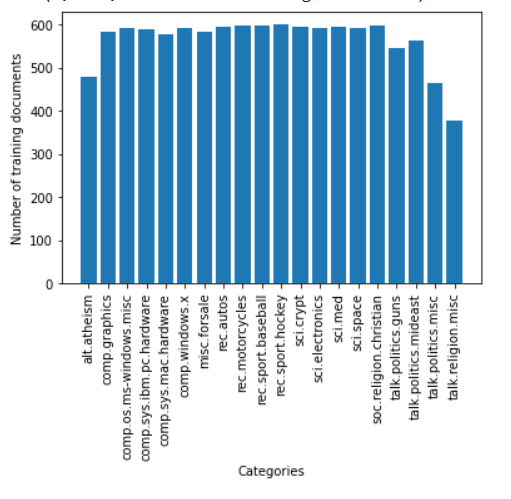
Da Yin 005453741

Amir Dhillon 005649833

**Introduction**

Statistical classification refers to the task of identifying a category, from a predefined set, to which a data point belongs, given a training data set with known category memberships. Classification differs from the task of clustering, which concerns grouping data points with no predefined category memberships, where the objective is to seek inherent structures in data with respect to suitable measures. Classification turns out as an essential element of data analysis, especially when dealing with a large amount of data. In this project, we look into different methods for classifying textual data.

**Q1 To get started, plot a histogram of the number of training documents for each of the 20 categories to check if they are evenly distributed.**



We see that the data set is already balanced and fairly evenly distributed, so we don’t need to do additional balancing work for the scope of this project. But overall, starting to work with a new data set - especially when it's a classification problem - seeing the distribution of data is recommended.

**Q2. Report the shape of the TF-IDF matrices of the train and test subsets respectively.**

The second step after checking the distribution of our data set, we begin to clean, extract, and process the data. We first select the categories we will be using for this project, then split them based on the existing sub-set to train, and test data set.

For the first part of the project, we will be doing binary classification, and right now, there are multiple classes to be labeled. We categorize these classes into the two classes “Computer Technology” & “recreational Activity”.

Then we follow these specs to extract features:

• Use the “english” stopwords of the CountVectorizer

This is one of the basic steps to ensure features we extract are essential and meaningful. By dropping the meaningless high-frequency words and punctuation, we are more likely to have words that are useful to training and testing the model, as well as reducing computation complexity by removing them.

• Exclude terms that are numbers (e.g. “123”, “-45”, “6.7” etc.)

We remove the numbers that aren’t going to be related to us understanding the meanings of text and words.

• Perform lemmatization with nltk.wordnet.WordNetLemmatizer and pos tag:

“*Stemming* usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes. *Lemmatization* usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the *lemma* .” ([Source](https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html)) In this project, after lemmatizing the list of words, we use Countvectorizer with min\_df=3 and “english” stop words to finish up preparing the data. The shape of x\_train and x\_test as follows :

Training dataset shape: (4732, 17406)

Testing dataset shape: (3150, 17406)

**Q3 Which one is larger, the norm score for NMF or LSI? Why is that the case?**

Both the Latent Semantic Indexing (LSI) & Non-negative Matrix Factorization (NMF) were used. They both have the purpose of minimizing MSE between the original data and the reconstruction from the low dimensional approximation.

Using k=50, which means that each document is reduced to map to a 50-dimensional vector. We use Truncated SVD from sklearn.decomposition and regular NMF, with the calculation of Frobenius norm as the performance measurement metric.

Frobenius Norm for LSI: 4152.90670825554

Frobenius Norm for NMF: 4186.58770421096

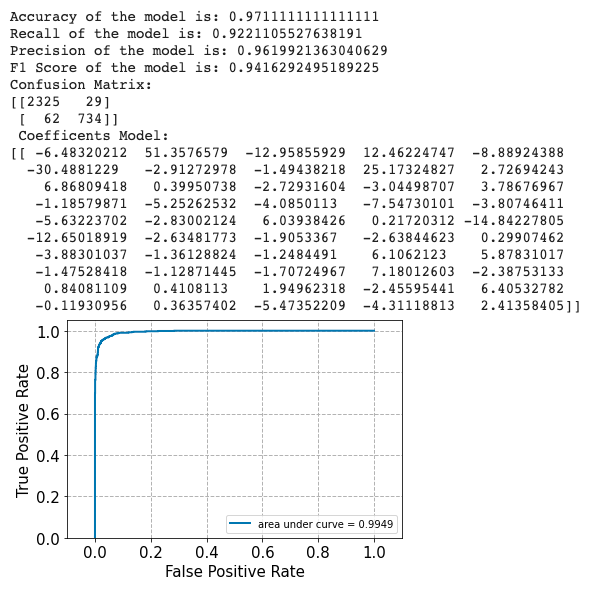
We see that the NMF has a slightly higher score, which means a higher residual error. Though NMF has great interpretability, its factorization is also not unique, therefore more constrained in the way that it has a local minimum problem. LSI filters out noises and reduces storage but at the same time, it also is hard to interpret and has an orthogonal restriction on basis vectors. They each have their strengths and weaknesses which is why the Frobenius norm(s) aren’t too far apart here.

**Classification Algorithms**

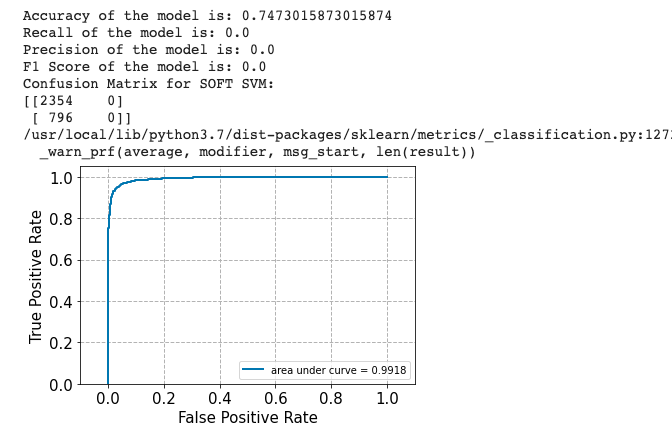
(Note: for binary classification & later on multiple classifications, we all use the LSI reduction x\_train\_LSI for training purposes, since the Latent Semantic Indexing method provided the best result)

**Q4 Hard Margin & Soft Margin**

**Hard Margin SVM Result: (Γ = 1000)**

****

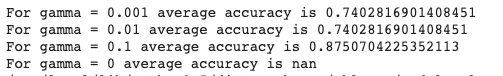
**Soft Margin SVM Result: (Γ = 0.0001)**

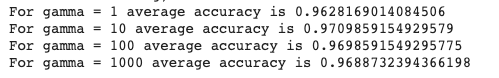


Soft margin SVM (**Γ**= 0.0001) has a significantly worse accuracy performance than Hard Margin SVM (**Γ**=1000). Hard Margin SVM strictly splits the data into two classes in binary classification vs. Soft Margin SVM might allow some outliers. Theoretically, soft margin SVM might have a better performance but since this is a binary classification between the related words for “computer technology” vs, “recreational activity”, the data seems very linearly separable in the way that words in either category are strongly associated with their category. This characteristic of data benefits hard margin SVM to have a higher accuracy score in this case. Additionally, our outputs reflect the above statements.

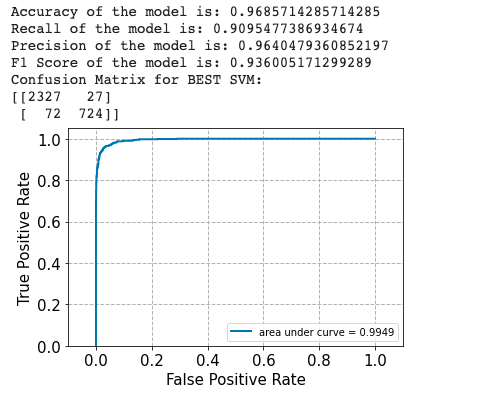
We use cross-validation (5-fold) in the array of **Γ** that ranges from 10^(-3) ~ 10^3 to print out the value of **Γ**, as well as its accuracy. We found the best **Γ** to be 100, and the performance metrics of **Γ**=100 is:

Best **Γ** = 100 with an average accuracy of 0.989859





Reporting the best **Γ** performance



**Q5 Logistic Classifier:**

Performance of Logistic Classifier without regularization:

Accuracy of the model is: 0.9415873015873016

Recall of the model is: 0.9415873015873016

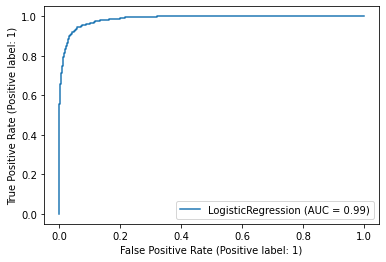
Precision of the model is: 0.9415873015873016

F1 Score of the model is: 0.9415873015873016

Confusion Matrix for Logistic Regression model

[[1462 98]

[ 86 1504]]



|  |  |  |
| --- | --- | --- |
| Regularization | L1 average training accuracy | L2 average training accuracy |
| 0.001 | 49.96% | 69.30% |
| 0.01 | 56.89% | 89.91% |
| 0.1 | 92.13% | 93.72% |
| 1 | 94.15% | 94.38% |
| 10 | 94.51% | 94.60% |
| 100 | 94.39% | 94.52% |
| 1000 | 94.41% | 94.42% |

Final Test Accuracy for L1 regularization:

Accuracy of the model is: 0.9406349206349206

Recall of the model is: 0.9406349206349206

Precision of the model is: 0.9406349206349206

F1 Score of the model is: 0.9406349206349206

Confusion Matrix for Logistic Regression model

[[1462 98]

[ 89 1501]]

Final Test Accuracy for L2 regularization:

Accuracy of the model is: 0.9415873015873016

Recall of the model is: 0.9415873015873016

Precision of the model is: 0.9415873015873016

F1 Score of the model is: 0.9415873015873016

Confusion Matrix for Logistic Regression model

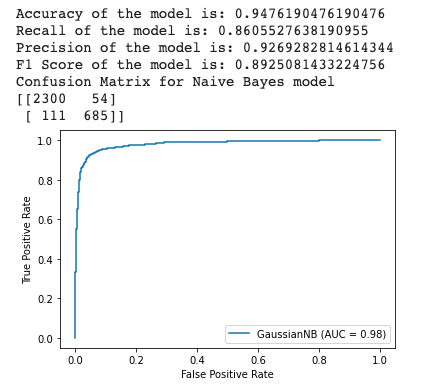
[[1459 101]

[ 83 1507]]

Regulazarition will reduce test error and overfitting. L2 regularization will reduce the impact of unimportant features. L1 regularization coefficients have some zero, and values ranges are larger.

SVM is trying to find the best margin between data points and the boundary, so it only cares about the closed points. Logistic Regression value more on overall distance from data to the boundary. In our test, SVM and Logistic regression performance did not differ that much, I think it is probably because the data is easy to train.

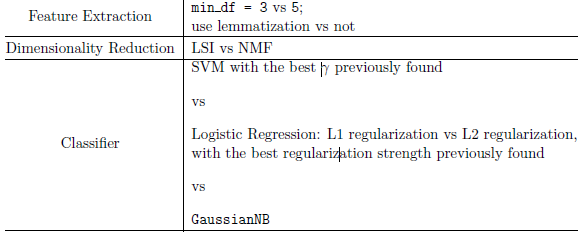
**Q6 Naive Bayes Classifier: GaussianNB classification**



The Gaussian Naive Bayes model is not the worst performing model in binary classification in this case, but it’s also not the best. GNB is part of the probabilistic classification which applies Bayes Rule to calculate the conditional probabilities. We will also use this model in the multiple class classification part, later on, to compare performance.

**Q7 : Grid search of parameters**

The pipeline includes a feature layer, a tfidf layer, a dimension reduction layer, and a classier. After running grid-search with the following options:



The best combination is

min\_df=3 for feature extraction

LSI as the dimension reduction method

LogisticRegression with L1 regulation, penalty =100

We consider the removing header and footer, and lemmatization as part of data pre-processing.

The test accuracy with the best model with the breakdown of different options to process data:

|  |  |  |
| --- | --- | --- |
|  | With Header and Footer | W/o Header and Footer |
| Lemmatization | 94.16% | 92.79% |
| W/o Lemmatization | 97.68% | 96.63% |

What was observed was that Lemmatization actually reduced the model accuracy. It could be caused by intensive word cutting. When we checked the lemmatized data, the words were barely recognizable, for example “from” was transformed to “fr”, and that is probably the reason that the test accuracy became lower.

The header and footer removed subject lines, which could contain important information of the content, so removing the header and footer caused a bit of accuracy decline as well.

**Q8 (a) Why are GLoVE embeddings trained on the ratio of co-occurrence probabilities rather than the probabilities themselves?**

The paper gave an example on word “ice” & “steam”, “Compared to the raw probabilities, the ratio is better able to distinguish relevant words (solid and gas) from irrelevant words (water and fashion) and it is also better able to discriminate between the two relevant words.” Overall, the relevance ratio between the two centered words, with its co-occurrence probability, is more appropriate being the word vector learning, instead of the probabilities themselves.

**(b) In the two sentences: “James is running in the park.” and “James is running for the presidency.”, would GLoVE embeddings return the same vector for the word running in both cases? Why or why not?**   
 GLoVe embeddings will likely not return the same vector for the word “running” in both cases, since GLoVe embeddings model uses the Overall Statistics and Local Context window. What likely would happen is that in the first sentence, “running” & “park” combination would show up significantly different from the “running” & “presidency” combination based on the GloVe methods used.  **(c) What do you expect for the values of…?**

The value of [Wife - Husband] would be roughly equal to [Queen - King], and they would both be bigger than [Queen - King- Wife + Husband].

Though this can be complicated, that wife-husband can be similar as in “married human” or “household” while they can be considered to be the opposite when genders assigned.

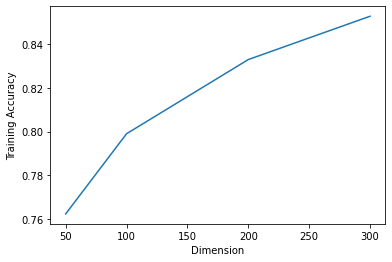
“The underlying concept that distinguishes man from woman, i.e. sex or gender, may be equivalently specified by various other word pairs, such as king and queen or brother and sister. To state this observation mathematically, we might expect that the vector differences man - woman, king - queen, and brother - sister might all be roughly equal. This property and other interesting patterns can be observed in the above set of visualizations.” - Original Author **(d) Given a word, would you rather stem or lemmatize the word before mapping it to its GLoVE embedding?**

Given a word - I would rather lemmatize the word for GLoVE embedding given lemmatization looks at the meaning of the word, and after applying lemmatization, we will always get a valid word. For GLoVE embedding where the meaning is important, I would rather lemmatize it if I can only pick one.

**Q9 : Binary classification task exercises**

Creating a weight matrix of dimensions vocab\_sizex300 (corresponding to the dimension of the GLoVe embeddings used) restricts the vector dimensions. Afterward, a matrix of embeddings for each word in the dataset is created. Keras sequential model was used for binary classification. In this specific model, we tried out a few different optimizers such as “adam”, “SGD”, “adamax”... and ended choosing “adam” as the higher accuracy optimizer.

**Q10 : GLoVE plot and accuracy**

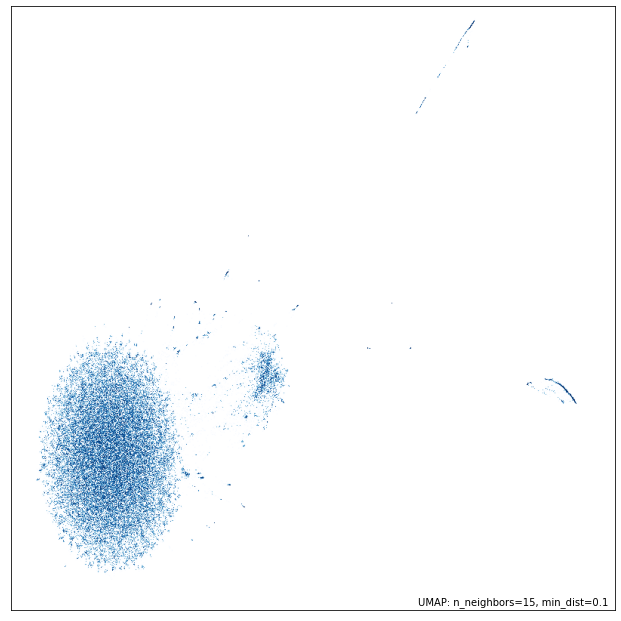


The plot shows the model accuracy increase as the dimension increases, and the peak training accuracy is around dimension = 300. It is not to be expected that a lower dimension has a lower accuracy score; since it’s known that algorithms may perform poorly in high dimensional data - “the Curse of Dimensionality”. Since here we are plotting the training accuracy and dimension, it might make sense why the accuracy increases as dimension increase (more information in data).

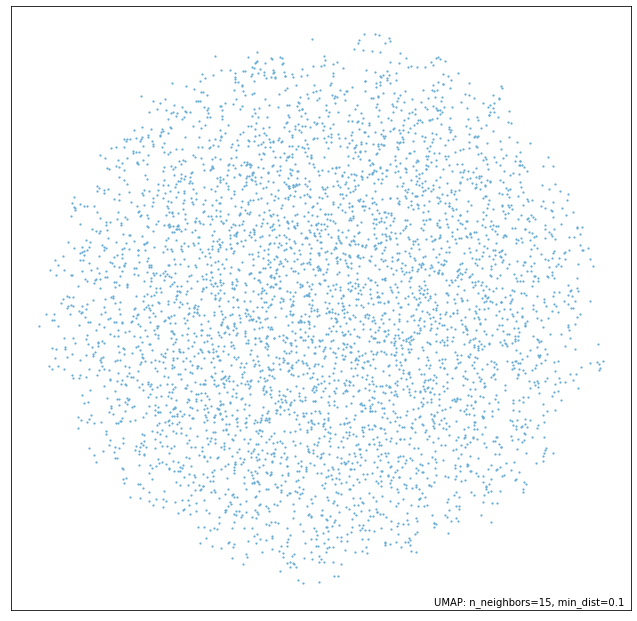
**Q11: Compare and contrast visualizations**

There’s minor clustering forming in the GLoVe embedding visualization (300 dimensions), looks completely random and distributed for the randomized vector of the same dimension.

UMAP plot fo GLoVe embedding (300 dimensions)



UMAP plot of the random vector

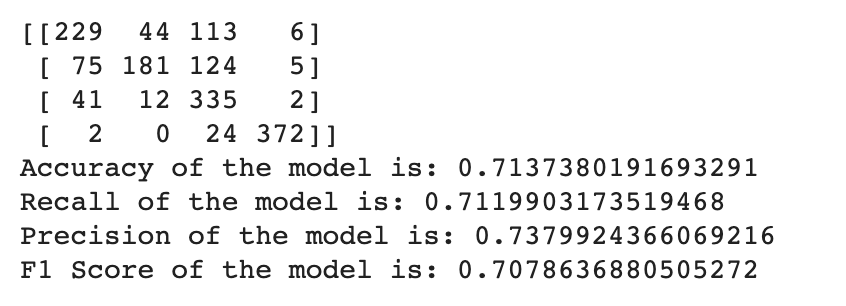


**Q12: Naıve Bayes classification and multiclass SVM classification (with both One VS One and One VS the rest methods described above)**

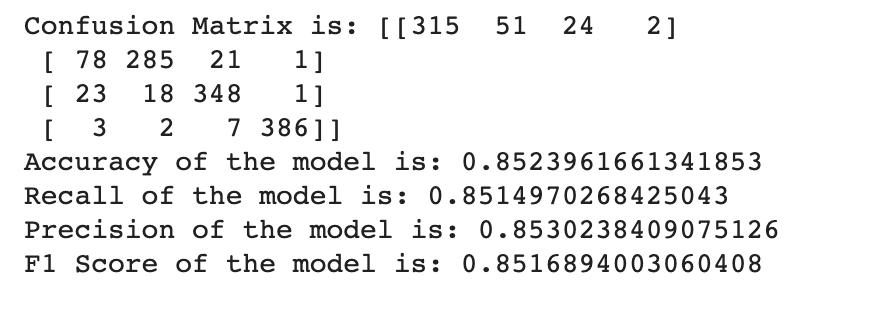
To re-prepare the dataset for multiclass classification, we are basically repeating the steps from questions 1-3, without categorizing these classes into two groups. After the extract, lemmatize, and reduce actions, we use the LSI reduced features to train and test the models. Almost all of the procedures were similar to previous parts, other than importing the OneVSOne and OnevsRest method, and making sure to have average = ‘macro’ in the performance report for recall, precision, and F1 Score. Marco average is the calculation of each labels’ unweighted mean, which works well in this case since the data distribution is fairly balanced to begin with.

Overall, Gaussian Naive Bayes has the worst performance out of the three; between Linear SVM OnevsOne and OnevsRest, they have very similar performance metrics, with OnevsOne just slightly better. In the code, you can view the two ways of us finding the best gamma: one way was by using a for loop to print out the accuracy score for gamma in the array - which we did to find the best gamma prior to the introduction of GridSearch; the other way was GridSearch.

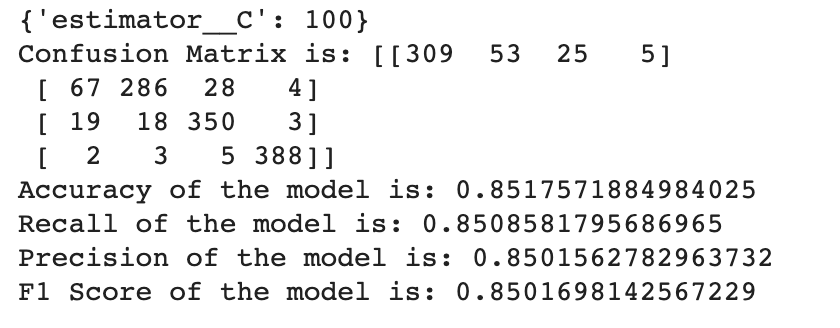
The three model’s performance and confusion matrix:

GNB Multiple class performance ****

One vs. One SVM multiple class performance

****

One vs. Rest SVM multiple class performance



**Works Cited**

**Ganegedara, T. (2020, December 4). *Light on Math ML: Intuitive Guide to Understanding GloVe Embeddings*. Medium. https://towardsdatascience.com/light-on-math-ml-intuitive-guide-to-understanding-glove-embeddings-b13b4f19c010.**

**Manning, C. (n.d.). Stemming and lemmatization. https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html.**

**Rul, C. V. den. (2019, September 24). *Understanding Word Embeddings with TF-IDF and GloVe*. Medium. https://towardsdatascience.com/understanding-word-embeddings-with-tf-idf-and-glove-8acb63892032.**

**Brownlee, Jason. “How to Use Word Embedding Layers for Deep Learning with Keras.” Machine Learning Mastery, 1 Feb. 2021, machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras.**