Compare and Contrast Modeling Multiclass and Binary Classification

Isabella Chittumuri

CUNY Hunter College

New York, NY

ABSTRACT

Understanding human emotion is a very difficult task, even for psychologists. However, there is always some bias that people can't look over. Machine learning can help eliminate that bias by just training on what it observes. We test two different types of dataset to see which one produces a higher accuracy score. The first one is testing a multiclass dataset with 26 emotions while the second one is testing on a binary class with only two emotions. We trained each dataset on these four models: Random Forest Classifier, Linear Support Vector Classifier, Naïve Bayes Classifier, and XGBoost. We compare and contrast the results of each model between the two datasets.

Keywords: google GoEmotions, deep learning, sentiment analysis, binary and multiclass

May 25, 2022

Research Advisor: Iordan Slavov

I. INTRODUCTION

A. Sentiment Analysis

Emotion expression and detection are important to human experience and social interaction. [1] With just a few words, we can express a large amount of complex emotions. Therefore, natural language processing (NLP) researchers developed sentimental analysis. Sentiment analysis is a type of NLP that analyzes online pieces of writing to determine the emotional tone they carry, removing human bias. [2]

B. Data Collection and Observations

GoEmotions is a corpus of 58k carefully curated comments extracted from Reddit, with human annotations to 27 emotion categories or Neutral, with comments extracted from popular English subreddits. [3] It contains 58,009 examples, 27 emotions including neutral, and a maximum sequence length of 30 in training and evaluation datasets. The emotion categories are: admiration, amusement, anger, annoyance, approval, caring, confusion, curiosity, desire, disappointment, disapproval, disgust, embarrassment, excitement, fear, gratitude, grief, joy, love, nervousness, optimism, pride, realization, relief, remorse, sadness, surprise. **Table 1** shows an example of the dataset.

Reddit is a large group of forums which registered users can talk about almost anything you can imagine, from news, to pop culture, to technology, to comics, to film, to literature The data contains reddit comments from 2005 to January 2019. The data was available in tsv format and there were no missing values. **Table 2** shows a general description of the data.

Table 1: Examples of Dataset

Sample #	Text	Class	ID
1	My favourite food is anything I didn't have to...	27	eebbqej
2	WHY THE FUCK IS BAYLESS ISOING	2	eezlygj
3	To make her feel threatened	14	ed7ypvh
4	Thanks. I was diagnosed with BP 1 after the...	15	efeeasc
5	Well that makes sense.	4	ef9c7s3
6	So glad I discovered that subreddit a couple m...	0	efbvgp9
7	Had to watch "Elmo in Grouchland" one time	27	edtjpv6

Table 2: Dataset Description

Description	Text	Class	ID
count	54263	54263	54263
unique	53994	782	54263

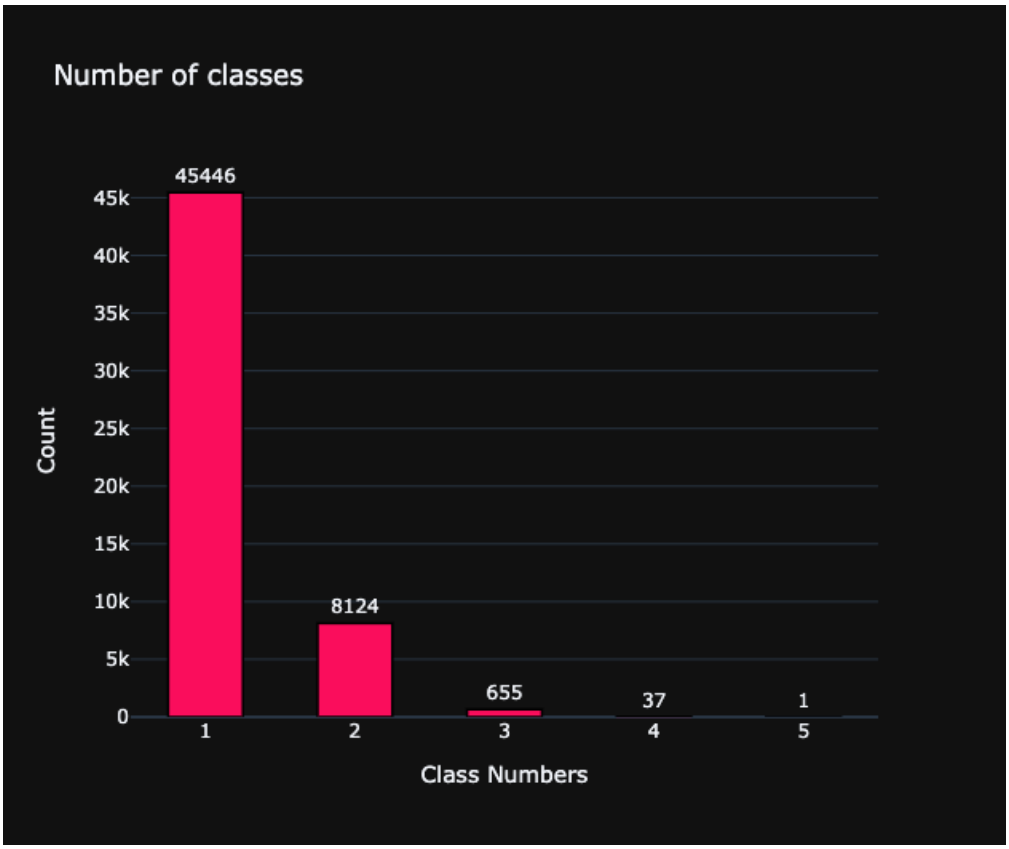
Top phrase	Thank you.	27	eebbqej
Frequency	15	16021	1

II. EXPLORATORY DATA ANALYSIS

A. Data Manipulation

First we combined the train, development, and test datasets into one date frame. We then had to see how much of the data has more than one class associated with it. To do this, we extracted a list of the classes as well as the length of the classes. We then plotted this data onto a bar graph seen on Figure 1. Since majority of the data only had one class, we removed those with two classes or more.

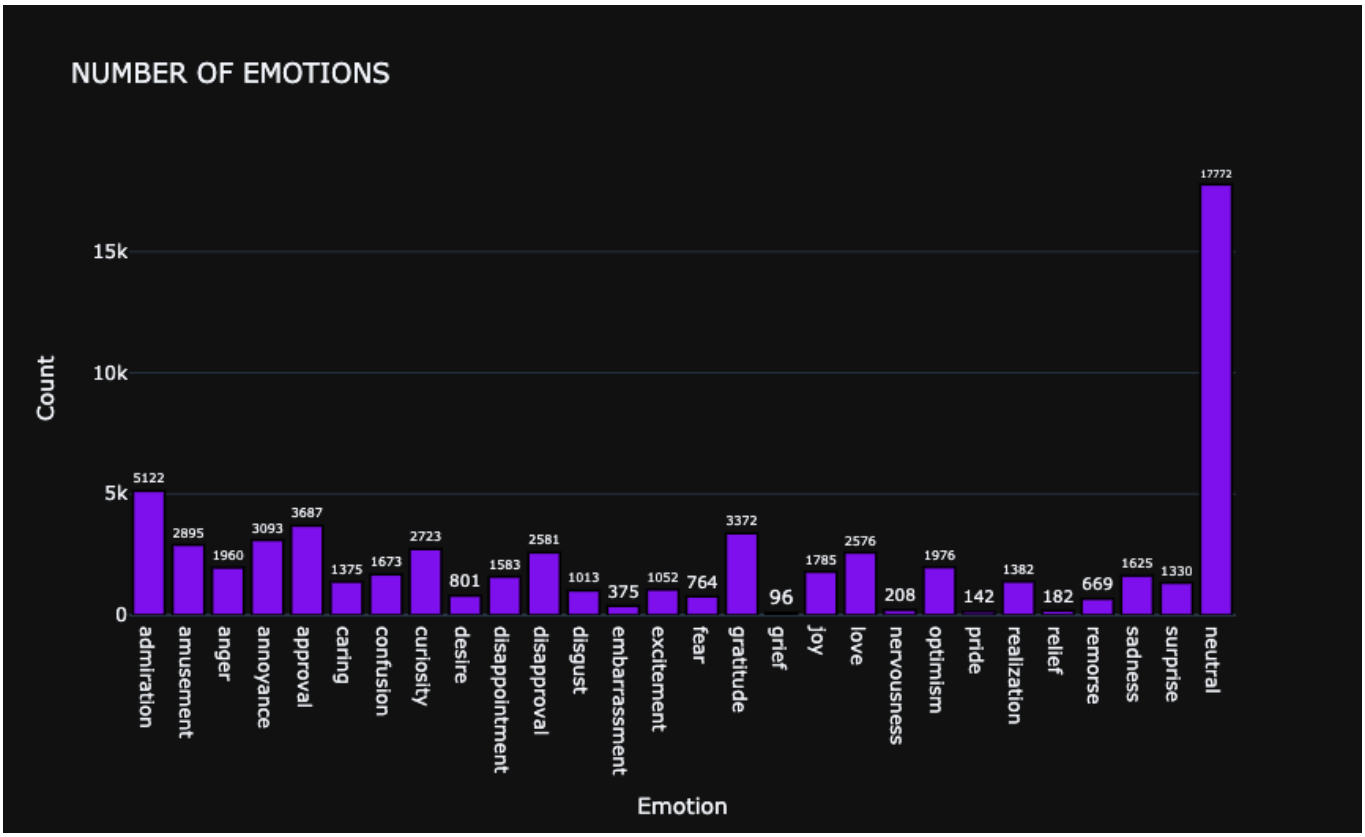
Figure 1. Bar graph of Number of Classes



Then we wanted to map the number of emotions in the dataset. We did this by downloading the emotions_map dataset, which associated each class ID with an emotion. Then

we extracted a text list of the classes in our dataset and mapped it to the emotions_map dataset. Figure 2 shows the distribution of emotions throughout our entire dateset.

Figure 1. Bar graph of Number of Classes



Since there was an uneven distribution of emotions, we decided to do two types of modeling: multi and binary classification. But first we had to do one-hot encoding to make transform the datasets into a vectorizer python can easily model. We used a TfidfVectorizer with max_features equal to 3000. TfidfVectorizer weights the word counts by a measure of how often they appear in the documents. We used Scikit-learn's train_test_split with a random_state of 42 to divide the data frame into X_train, X_test, y_train, y_test. The test size contains 33 precent of the data.

III. MODELING

For multiclass, we removed the emotion to create a more even distribution within the other 26 emotions. For binary, we combined all emotions excluding neutral into one class and kept neural as the second class.

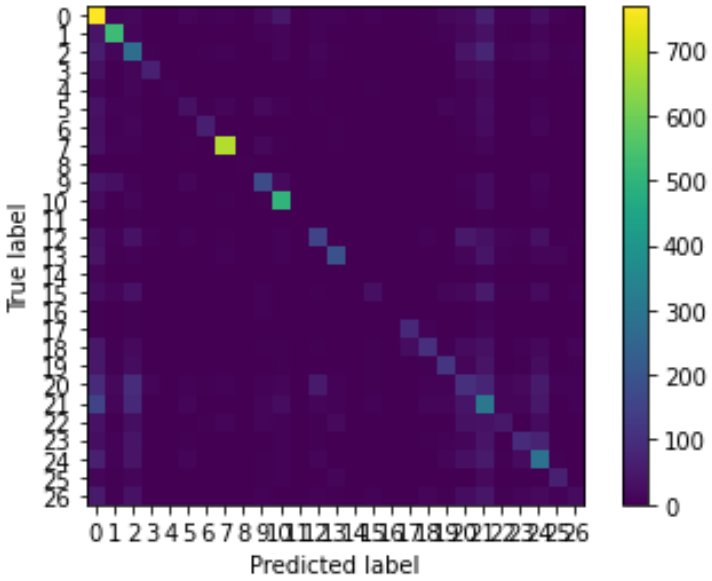
A. Multi-Classification

For multiclass, we removed the emotion neutral to create a more even distribution within the other 26 emotions.

1. Random Forest

The random forest is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree. [4] We used Scikit-learn's Random Forest Classifier and fit the model using the X_train and y_train datasets. Then we predicted the model on the test set. This gave us an accuracy score of 48.55 percent. This means that the model preforms slightly less than chance. We graphed the confusion matrix of predicted and true values of the test dataset. Figure 3 shows this confusion matrix. The diagonal line of the confusion matrix shows what the model predicted was correct with its frequency of accuracy seen by the color gradient. Anything with some saturation outside of that line is what the model predicted to be true was actually false.

Figure 3. Multiclass Random Forest Confusion Matrix



We also printed the classification report of the predicted and true values, seen in Figure 4. In terms of the classification report, precision is the ability of a classifier not to label an instance true that is actually false, or vice versa. Recall is the ability of a classifier to find all stable instances or all unstable. F1-score is the weighted harmonic mean of precision using both Precision and Recall; it's used to see if the dataset has an imbalanced distribution of the different classes. Support is the number of actual occurrences of each class in the *y_test* dataset.

Imbalanced support in the training data may indicate structure weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing. Support doesn't change between models but instead diagnoses the evaluation process.

In terms of the averages, macro average is computed using each column's results and returns the average without considering the weights for each class. This can result in a bigger penalization when our model does not perform well with the minority classes, which is exactly what we want when there is imbalance. Weighted average is computed using each column's results and returns the average considering the weights for each class. This favors the majority class, which is want we usually don't want. In our case, both the macro and weighted averages were relatively the same for all columns

Figure 4. Multiclass Random Forest Confusion Matrix

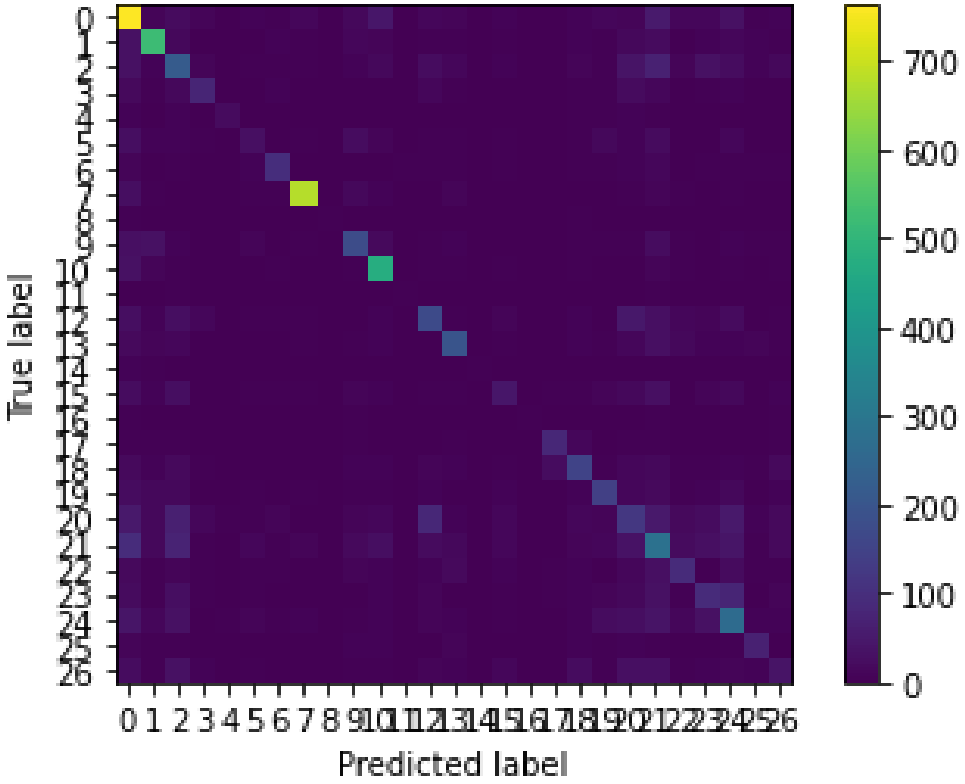
	precision	recall	f1-score	support
0	0.44	0.71	0.54	1079
1	0.76	0.75	0.75	700
10	0.32	0.48	0.38	575
11	0.60	0.32	0.42	210
12	0.43	0.08	0.13	78
13	0.25	0.17	0.20	195
14	0.63	0.35	0.45	185
15	0.89	0.86	0.88	785
16	1.00	0.27	0.42	15
17	0.55	0.50	0.52	355
18	0.71	0.86	0.78	583
19	0.20	0.04	0.07	23
2	0.49	0.37	0.42	418
20	0.69	0.54	0.60	361
21	0.00	0.00	0.00	29
22	0.35	0.13	0.19	249
23	0.50	0.06	0.11	32
24	0.64	0.64	0.64	138
25	0.54	0.31	0.39	338
26	0.53	0.37	0.44	314
3	0.21	0.17	0.19	621
4	0.26	0.38	0.31	788
5	0.34	0.18	0.23	284
6	0.41	0.26	0.32	351
7	0.39	0.49	0.43	576
8	0.54	0.45	0.49	152
9	0.21	0.08	0.11	277
accuracy			0.49	9711
macro avg	0.48	0.36	0.39	9711
weighted avg	0.49	0.49	0.47	9711

We also computed the top_k_accuracy_score. This metric computes the number of times where the correct label is among the top k labels predicted. [5] We set k to equal 2, which means that the model gets two changed to try to predict the correct label. This score resulted in 60.27 percent. This means that the model did slight better than chance. Logistically speaking since people in general have a hard time accurately predicting someone's emotions, but does much better after given a few tries. It overall preformed the same as the random forest.

2. Linear Support Vector Classifier

The linear support vector classifier is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier. It has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples. [6] Similar to random forest, we trained the model using Scikit-learn's LinearSVC on the training dataset. This model yielded an accuracy score of 50 percent. It's confusion matrix is shown in Figure 5.

Figure 5. Multiclass LinearSVC Confusion Matrix



3. Naive Bayes Classifier

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. [7] We used Scikit-learn's Gaussian Naïve Bayes

with a variance smoothing of 1e-5. Because of the structure of our dataset, we have to fit the model by making X_train dense. This resulted in an testing accuracy of 10.89 percent. Using top_k_accuracy_score of k=2, we got an accuracy score of 15.89 percent.

4. Extreme Gradient Boosting

Lasty, we tried Extreme Gradient Boosting (XGBoost). XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. Gradient boosting is a supervised learning algorithm, which attempts to accurately predict a target variable by combining the estimates of a set of simpler, weaker models. However, once we fit the model, it only resulted in an accuracy score of 10.89 percent.

B. Binary Classification

To do a binary classification, we combined all emotions excluding neutral into one class and kept neural as the second class. We labels neutral as 0 and all other emotions as 1. Table 3 describes different characteristics of this modified dataset.

Table 3: Dataset Description

Description	Text	Class
count	54263	54263
unique	53994	2
Top phrase	Thank you.	1
Frequency	15	16021

1. Random Forest

Using the same method for the multiclass dataset, we fit the random forest classifier model and got an accuracy score of 55.88 percent. We printed the confusion matrix and classification report which is show in Figure 6 and 7. Based on these metrics, it seems like the model predicted true values slightly better than that of the multiclass model. When we computed the top_k_accuracy_score with k=2, we got an accuracy of 85.64 percent.

Figure 6. Binary Random Forest Confusion Matrix

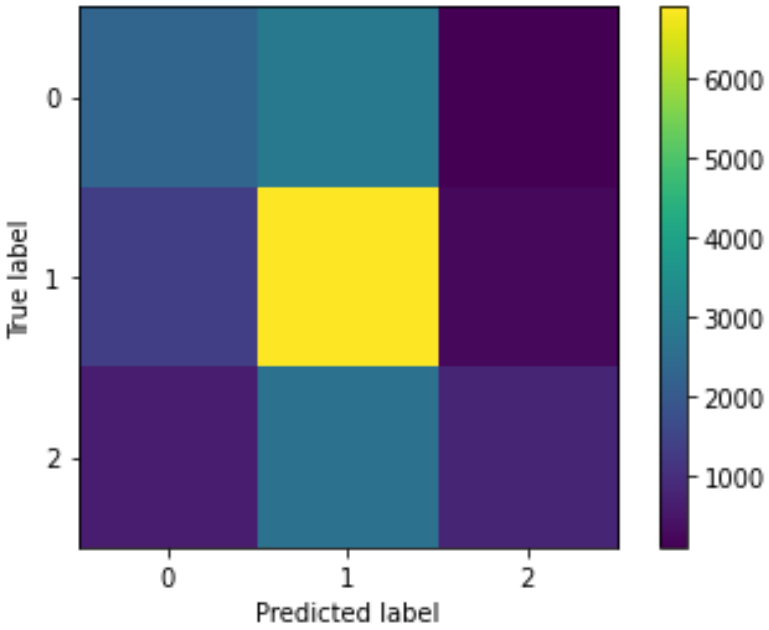


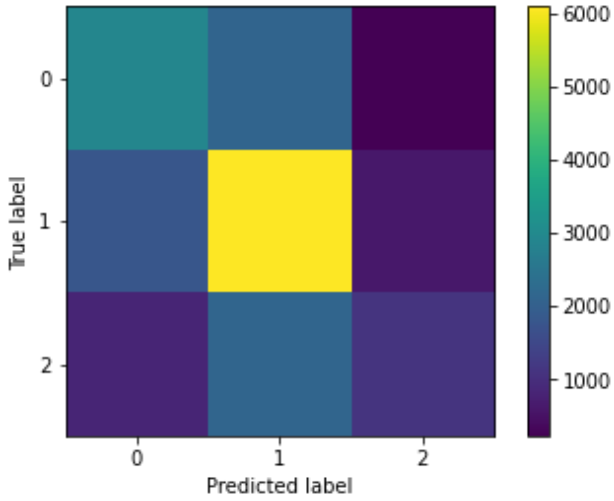
Figure 7. Binary Random Forest Classification Report

	precision	recall	f1-score	support
0	0.54	0.44	0.48	5290
1	0.55	0.81	0.66	8517
nan	0.68	0.20	0.30	4100
accuracy			0.56	17907
macro avg	0.59	0.48	0.48	17907
weighted avg	0.58	0.56	0.53	17907

2. Linear Support Vector Classifier

When modeling this binary class dataset on the linear support vector classifier, we got an accuracy score of 56.68 percent. It does very similar the random forest model. Figure 8 shows the LSV confusion matrix. It seems like both models highly predict emotions of anything other neutral. The binary version of this model did much better than that of the multiclass model.

Figure 8. Binary LinearSVC Confusion Matrix



3. Naive Bayes Classifier

We again used the Scikit-learn's Gaussian Naïve Bayes classifier on the training set. In training the model, we used a variance smoothing of 1e-5 and transformed X_train to dense. This yielded a testing set accuracy score of 41.20 precent. Using k=2, we calculated a top_k_accuracy_score of 73.59 percent.

4. Extreme Gradient Boosting

Lastly, we trained on the same XGBoost model that we used on the multiclass dataset. We got an accuracy of 41.20 percent. This model did much better that of the multiclass, about 30 percent more. This could be due to the fact that the model doesn't have to train on many different emotions that have similar connotations.

IV. CONCLUSION

Emotions are generally hard to detect for humans. So in creating and training a machine learning model to do just that will take a few trials. The binary random forest classifier preformed only 6 percent higher than that of the multiclass. This was also the same case for the linear support vector classifier. However, the binary Naïve bayes did significantly higher within the top_k_accuracy score. This is due to the fact that this type of classifier uses the bayes theorem which is strongly suited for binary classes. XGBoost classifier also did better when training only on binary classes. This makes sense since the machine learning limits how many variances there are in training, when training only on binary classes. Future research can include testing more modeling and seeing if there are some pre-trained models that could handle the complexities of many emotions.

V. REFERENCES

[1] *Goemotions: A dataset of fine-grained emotions - arxiv.org*. (n.d.). https://arxiv.org/pdf/2005.00547.pdf

[2] Tom, Tom Good food, & food, G. (2022, May 18). *What is sentiment analysis? an ultimate guide for 2022*. Brand24 Blog. https://brand24.com/blog/sentiment-analysis/

[3] Google-Research. (n.d.). *Google-Research/goemotions at master · google-research/google-research*. GitHub. https://github.com/google-research/google-research/tree/master/goemotions

[4] Yiu, T. (2021, September 29). *Understanding random forest*. Medium. https://towardsdatascience.com/understanding-random-forest-58381e0602d2#:~:text=The%20random%20forest%20is%20a,that%20of%20any%20individual%20tree

[5] *Sklearn.metrics.top_k_accuracy_score*. scikit. (n.d.) https://scikitlearn.org/stable/modules/generated/sklearn.metrics.top_k_accuracy_score.html#:~:text=Top%2Dk%20Accuracy%20classification%20score,(ranked%20by%20predicted%20scores)

[6] *Sklearn.svm.LinearSVC*. scikit. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

[7] *1.9. naive Bayes*. scikit. (n.d.). https://scikit-learn.org/stable/modules/naive_bayes.html