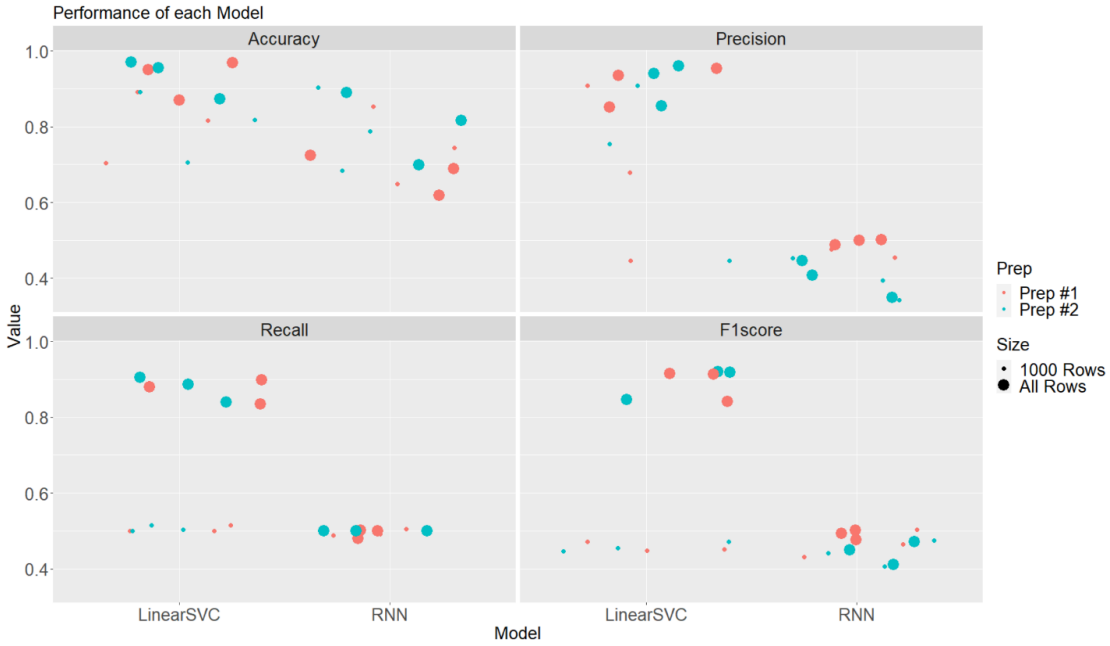
Part 1: Text Classification

To develop this task, after combining the 4 different parameters shown below, we created 24 (3x2x2x2) different models, and our task is to identify which parameters improve the performance of the prediction of each label:

1. Labels:
    a. InfoTheory
    b. CompVis
    c. Math
2. Preprocessing Methods:
    a. Method #1
    b. Method # 2
3. Size of training dataset:
    a. All rows
    b. First 1000 rows
4. Models:
    a. LinearSVC – Statistical model
    b. RNN

Once we performed the text classification in our Jupyter Notebook, we plot the results below



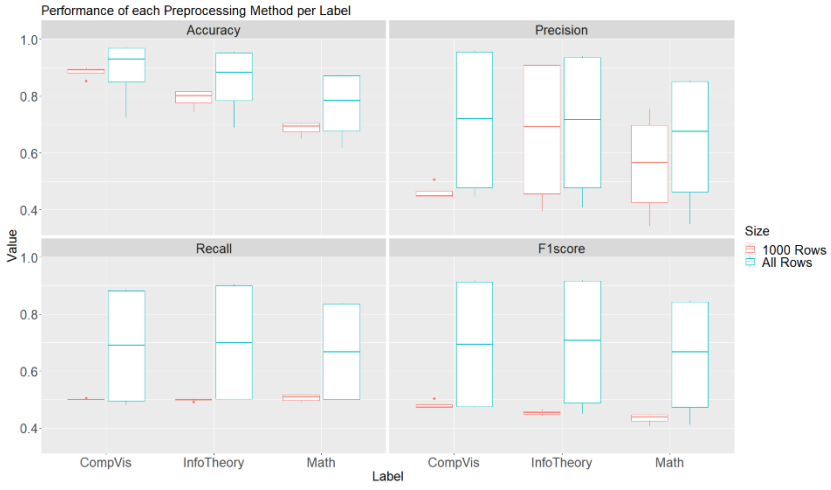
Ignoring the label, we wanted to get some insights about the models, at this point a label is just an input for 8 different models a with this one we got some interesting insights:

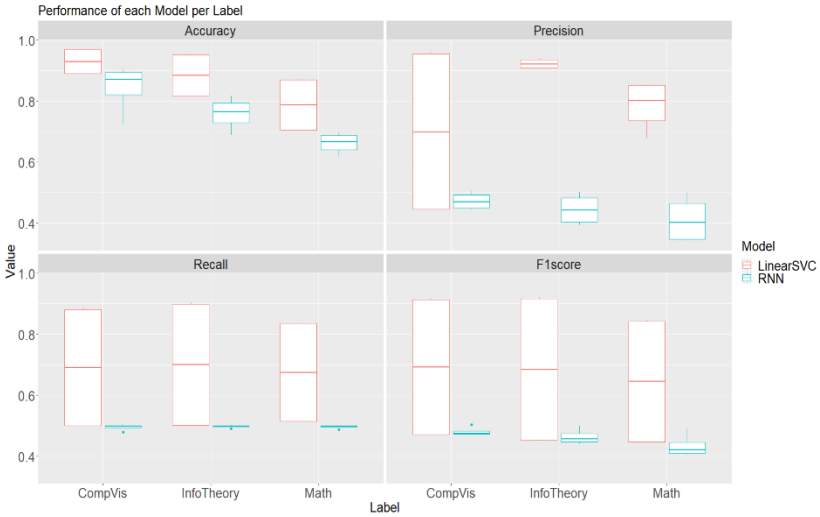
1. In all the metrics the LinearSVC model performed way better than the RNN. By looking just at the accuracy of the best RNN model, it is not better than 4 different LinearSVC models, and comparing with the other three metrics the RNN models are just as good as the worst LinearSVC models. Therefore, by just comparing the models we can identify that it was better the LinearSVC and the RNN model.
2. From the plot above we can identify that the performance of each Preprocessing Method is quite similar. There is always a red dot next to the green one, and it seems that the performance changes due to other factors different than the preprocessing method.
3. The size of the training dataset has also a great impact on the performance of the models, in metrics like Recall and F1-score, there is a big gap between the models that used the complete training dataset compared with the models that only used the first 1000 rows. One of the reasons for this difference is that even though all the labels are unbalanced in the whole dataset, in the first 1000 it is even more unbalanced. Leading us to worse results in the prediction. This gap can also be observed in metrics like Accuracy and Precision, even though it is not that big, that is why we considered that it is better to train the models with the bigger dataset in this case All Rows

To support the insights mentioned before, and have some insights regarding the labels, we have some other plots where we grouped the labels, to show the performance of each one of them compared with the other parameters: Preprocessing Method, Size and, Model

From this plot, it can be seen with more details that the performance due to the preprocessing method does not change so much, and in some metrics, it is better the method #1 and in other the method #2.
We can also observe that each label has a similar Recall and F1score in different models. But it does have a difference in the Accuracy, leading us to think that it is easier for different models to predict the CompVis labels and it is harder to predict Math

Like we observed in the last plot, the accuracy of each Label leads us to think that is easier for the models predict the label CompVis compared with InfoTheory and Math.
Besides, we can observe that in every case the performance of the models trained with the total of rows of the dataset is better than the models trained with only the first 1000 rows. The gap between models is big, therefore the size is one of the most important parameters for the models





Finally, with the plot, we can confirm what we have been saying. The easiest Label to predict is CompVis and the hardest one is Math.

Moreover, the performance of the LinearSVC model is better in all cases than the RNN model. However, the metrics of the RNN are more constant than the LinearSVC because the boxplots have a smaller range compared with the wide boxplots of the LinearSVC

In conclusion, the best model in this case is the one with these parameters: LinearSVC model, trained with all rows and with any Preprocessing Method chosen.

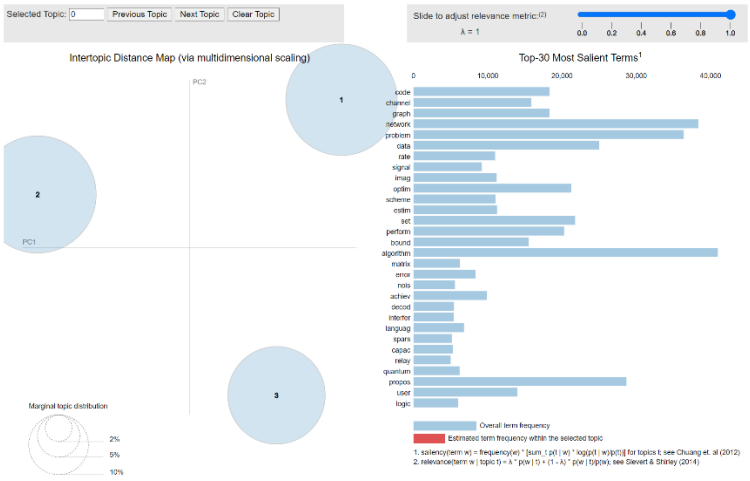
The performance of this optimal model is better to predict the value of CompVis

Part 2: Topic Modelling

In this task, we combined 2 different parameters shown below, to create 4 different topic classification with LDA model. The parameters are: Size of the training dataset and number of topics

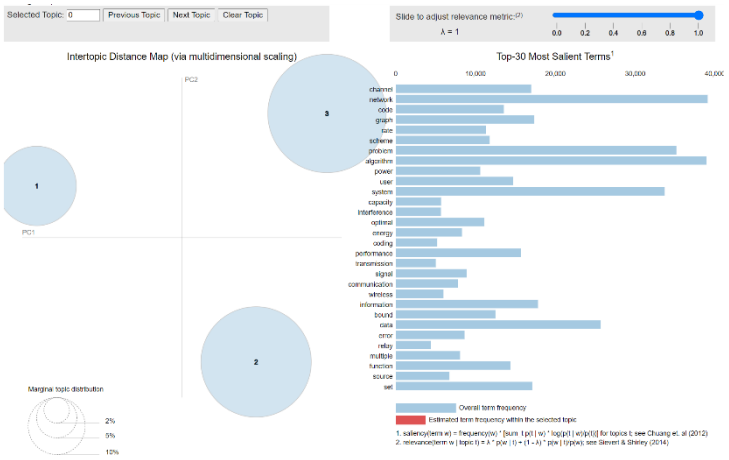
Model 1

In this model we used Preprocessing Method #1, that it is explained in the Jupyter notebook but it is relevant to mention that is this method we removed the stop words and divided the abstracts in 3 different topics.



From the following graph we can identify that there are 3 topics completely different from each other. As the distance between them is big and the circles are in different positions without being overlapped, therefore the topics does not have information in common. Also, we can identify that each topic has approximately the same number of documents as the size of each of them is similar.

Model 3



Like we did with Model 1, Model 3 has three different topics too, but in this case, we used the Preprocessing method #2, explained in the Jupyter notebook, but it is relevant to mention that this does have stop words, adding some noise to the model. That is why we can observe even though is seems to be the same Model 1, three completely different topics without overlapping, in this case there is one topic with half of the documents than the other two.

It is already known that the abstracts are classified into three different topics: InfoTheory, CompVis, and Math. We selected 3 different topics in these two different models to find if it is easy to classify again the documents in these specific three different topics. And by just looking at the top 5 words of each topic it seems hard to get the same topics:

```
0: 0.016*"problem" + 0.015*"algorithm" + 0.011*"graph" + 0.011*"set" + 0.009*"gener"
1: 0.015*"network" + 0.012*"data" + 0.012*"model" + 0.011*"system" + 0.010*"base"
2: 0.014*"code" + 0.013*"channel" + 0.011*"optim" + 0.009*"propos" + 0.009*"algorithm"
```

Model 1

```
0: 0.015*"channel" + 0.011*"code" + 0.010*"network" + 0.009*"rate" + 0.009*"scheme"
1: 0.014*"algorithm" + 0.013*"problem" + 0.008*"graph" + 0.008*"a" + 0.008*"be"
2: 0.011*"network" + 0.010*"a" + 0.009*"system" + 0.009*"data" + 0.008*"model"
```

Model 2

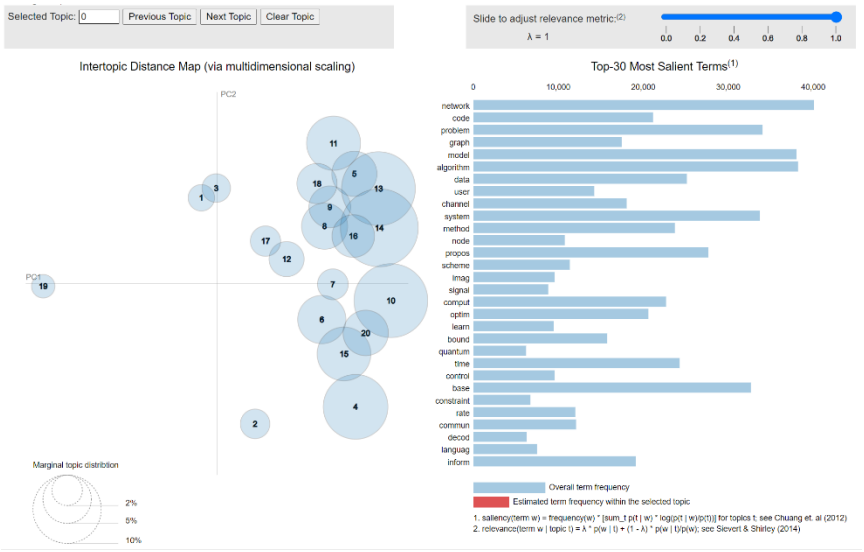
Topic 1: Graphic Algorith Problems.
Topic 0 of Model 1 and Topic 1 of Model 3

Topic 2: Code of Network Channel.
Topic 2 of Model 1 and Topic 0 of Model

Topic 3: Network Data Model System
Topic 1 of Model 1 and Topic 2 of Model

Model 2

In this model, as we did with model 1 we used Preprocessing Method #1, removing stop words but in this case, we divided the abstracts into 20 different topics.

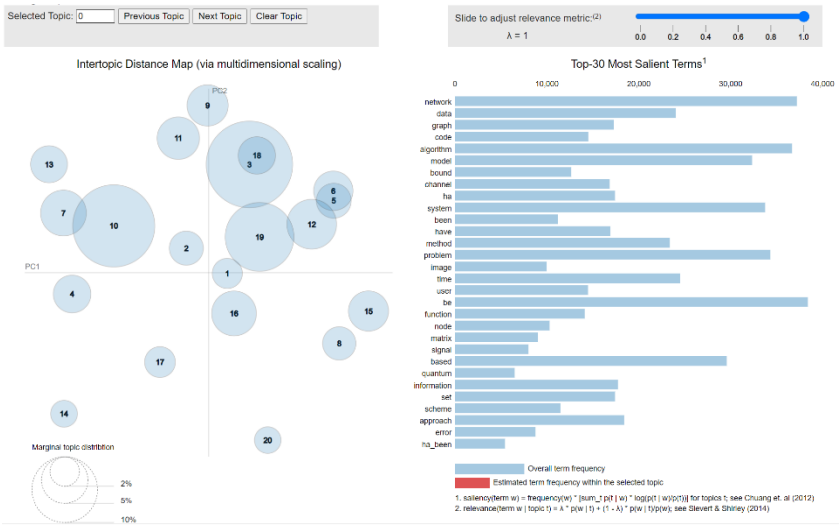


From this graph can be seen that many topics are related to each other because they are close to each other and some of them are overlapped sharing information between topics.

However, there are three different groups, one in the bottom right corner with 7 topics, one in the top right corner with 1o topics, and finally the left side of the graph with 3 topics.

Model 4

Classifying the same abstracts in 20 different topics with a different Preprocessing Method, that has stop words changes the result.



In this plot, we do not observe any pattern between topics as we could observe before. But in this case, we can see that there are topics embedded into others, like topic 18 in topic 3, leading us to think that maybe topic 18 is a subtopic of 3.

To name each topic when we have 20 different of them is hard, because each topic shares many words with other topics, because of the relationship between them. That is why we do not bring the top 5 words of each topic here (in the jupyter notebook).

Finally, it is relevant to say that, even though we should have needed more experiments to confirm this, it seems that the stop words add more noise when it comes to creating more topics. In the first two models that we only had 3 different topics, the position and the size of them remain almost the same. But when we did the same with 20 different topics it seems to be completely different abstracts being classified as the position and the pattern of the topics changed completely