FIT3164 Data Science Project

Semester 2, 2021

Final Project Report

Scope : *Prediction in performance of the information retrieval systems using a classification algorithm*

Group name : FIT3163_MA_2

Group members : Ong Jeng Quan, Ong Meng Lap, Ong Meng Chuan

Project supervisor : Dr. Prabha Rajagopal

Tutor's name : Ms Kamala

Due date : 22 October 2021

Date of submission : 22 October 2021

Word count : 7382 words

Table of contents

1 Introduction 4

2 Background 4

2.1 Literature Review 4

2.1.1 Literature Review introduction 4

2.2 Literature Review Content 5

2.2.1 TREC-COVID 5

2.2.2 Related work 6

2.3 Literature Review Conclusion 6

3 Outcomes 7

3.1 Results Achieved 7

3.2 How are Requirements met 12

3.3 Justification of Decisions Made 14

3.3.1 Choice of dataset 14

3.3.2 Creation of class label 16

3.3.3 Evaluation metrics used 17

3.3.4 Choice of machine learning algorithm 18

3.3.5 Hyperparameter tuning using Random Search 18

3.4 Limitations of Project Outcomes and Possible Improvements 20

3.4.1 Overfitting of classification model 20

3.4.2 Participants involved in usability testing 21

3.4.3 Design of web application 21

4 Methodology 23

4.1 System Design 23

4.2 Data Preprocessing 25

4.3 Data Modelling 26

5 Software Deliverable 27

5.1 Summary of software deliverable 27

5.1.1 What is delivered 27

5.1.2 Sample screenshots of usage 29

5.2 Summary and discussion of software qualities 33

5.2.1 Robustness 33

5.2.2 Security 34

5.2.3 Usability 35

5.2.4 Scalability 35

5.2.5 Documentation and Maintainability 35

5.3 Sample Source code 35

6 Critical Discussion 36

6.1 Deviations from initial project proposal 36

6.1.1 Hosting web application online 36

6.1.2 Project scheduling 36

7 Conclusion 37

8 Appendix 38

8.1 Sample Code 1 38

8.2 Sample Code 2 39

8.3 Sample Code 3 40

9 References 41

10 Annex 42

1 Introduction

Ever since the Coronavirus Disease 2019 (COVID-19) became a public health issue, there has been a rapid growth of scientific publications from various sources regarding COVID-19. This has led to a large corpora of scientific publications for researchers in the medical field to examine it. Given that conventional reading methods are challenged, automated text mining techniques that involve searching, reading, and summarizing articles have been performed to handle this enormous amount of information (Wang & Lo, 2021). The application of text mining has managed to solve this problem by retrieving relevant documents along with the usage of some popular text mining algorithms. Moreover, information retrieval (IR) systems were also used to manage the access to the large corpora of publications (Roberts et al., 2020).

In addition, there has also been an increase in inaccurate or partial information regarding COVID-19 being spread widely across the globe, which has resulted in an information crisis. Therefore, in order to retrieve articles which are only relevant to the search query given by researchers in the medical field, a TREC-COVID challenge has been organized to evaluate information retrieval (IR) systems (Chen & Hersh, 2021). The main objective competition is to reduce the widespread of irrelevant articles being retrieved from large amounts of publications which are released daily through an IR system. From this challenge, we have realized that the IR system of each participating team has their own methods and algorithms in determining the relevance of the articles being retrieved. However, a gap exists in the current work where it does not reveal how the IR systems are considered as a low or high performing IR system.

Therefore, this has led us to a search in identifying what are the attributes of an IR system that correlates to the performance of an IR system using a classification model, which will give us some underlying reasoning that contributes to the performance of an IR system throughout this project.

2 Background

2.1 Literature Review

2.1.1 Literature Review introduction

The COVID-19 pandemic has caused a rapid growth of scientific information at an unprecedented rate, where there are more than 50 000 scientific publications regarding COVID-19, that have been published ever since the beginning of 2020 (Wang & Lo, 2021). Additionally, several hundreds of new COVID-19 publications are also being published daily by the Semantic Scholar at the Allen Institute for AI as well. Consequently, this has eventually led to an information overload which makes it difficult for researchers or experts to carry out research to access the important information regarding COVID-19. Therefore, a TREC-COVID challenge that is supported by National Institutes of Standards and Technology (NIST) was introduced to

evaluate the capabilities of an information retrieval (IR) system in retrieving relevant articles for a query given by the user (Soni & Roberts, 2021).

2.2 Literature Review Content

2.2.1 TREC-COVID

Furthermore, the COVID-19 Open Research Dataset (CORD-19) which is currently known to be one of the earliest yet largest corpora was used as a dataset by each group of participants who have participated in the TREC-COVID competition (Wang & Lo, 2021). Besides, both well-known technology companies Google and Amazon have also developed their own IR systems using the similar CORD-19 dataset which are "COVID-19 Research Explorer3" and "CORD-19 Search2" respectively (Soni & Roberts, 2021).

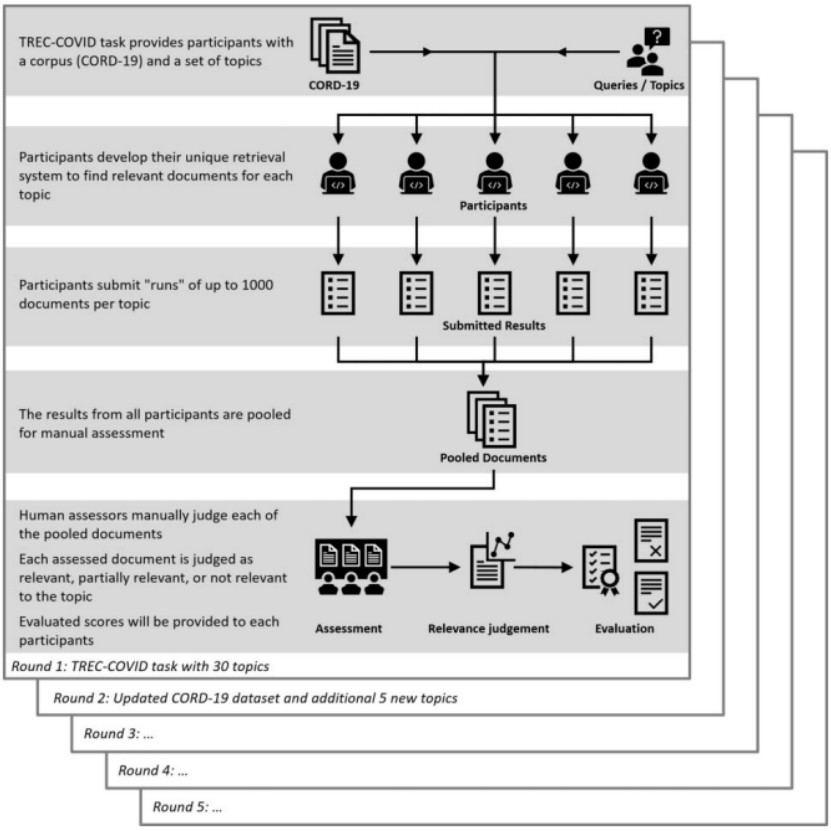


Figure 1. *Workflow of the project of TREC-COVID. (Roberts, K., Alam, T., Bedrick, S., Demner-Fushman, D., Kyle, Soboroff, I., Voorhees, E., Wang, L. L., & Hersh, W. R., 2020)*

Referring to Figure 1 above, the TREC-COVID competition pool results obtained from the unique IR systems submitted by each group of participants. In addition, a collaboration with a popular online community of data scientists' practitioners such as Kaggle has also been done to improve the participation rate of this competition (TREC-COVID Organizers, n.d.). The results obtained from each IR system will then be assessed manually by human assessors and are

labelled based on the relevancy of the document to the topic, with 0 being irrelevant, 1 being partially relevant and 2 being relevant.

However, the current work produced by TREC-COVID does not provide us information regarding features of an IR system that correlates to the performance of an IR system which is the main research gap that we have identified from this competition that has been organized. In other words, we have not identified if there are any specific methods or algorithms that have been used by an IR that affects the performance of an IR system. As we understand that different architectures of an IR might influence the performance of an IR system in the field of IR.

2.2.2 Related work

Based on the research that we have done, previous studies have found that the performance of an IR system has been improved significantly through fine-tuning ranking systems along with the usage of relevance judgements (Chen & Hersh, 2021). In addition, Chen and Hersh (2021) have also found that query formulation plays a significant role for a successful search through an IR system. For example, participating groups that use the query and question fields in generating input query in their IR system have eventually performed better as compared to groups that use the narrative proportion of a topic to generate an input query. Besides that, we have also found out that each IR system of each participating groups in the TREC-COVID challenge has their own sets of methodology and algorithms to determine the relevancy of articles such as 'Ranking', 'BERT', 'Top-k' and 'TF-IDF'. This has given us an idea that the usage of certain algorithms or methodology within an IR system could be predictors which are believed to correlate to a certain range of performance for the IR system where there is currently no published research that focuses on the identifying if the usage or certain algorithms or methodology is deemed to affect the performance of an IR system to the best of our knowledge and research that we have done.

2.3 Literature Review Conclusion

Therefore, in order to address the gap from the current work produced from TREC-COVID. This leads us to the motivation of our research in identifying whether certain usage of methodology or algorithm within an IR system tend to correlate to the performance of an IR system which will result in either a low or a high performing IR system effectively using a classification model. Additionally, the final product that will be delivered will not only contribute to the researchers from the medical field, specifically on the matters of COVID-19 by providing information regarding performance of an IR system, but also provide further opportunities for future work regarding TREC-COVID and related test collections that could improve the quality of the articles retrieved in the future.

3 Outcomes

3.1 Results Achieved

The final product of our project that will be delivered is a web application that has been implemented with the core features which has allowed user to predict the performance of an IR system effectively through the classification model that we have implemented using a Random Forest algorithm in order to meet the project requirements. A screenshot of the header and the main section of the website is shown below in Figure 2. is a web application that has been implemented with core features - that allows user to.

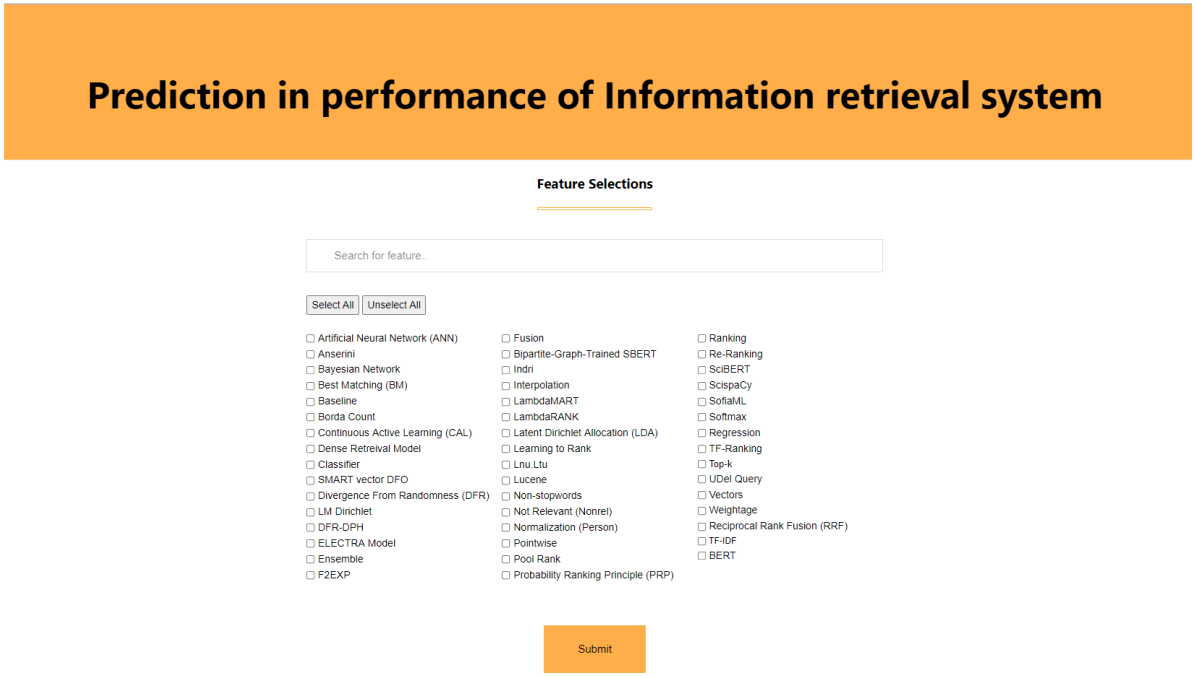


Figure 2. *Main section of the web application*

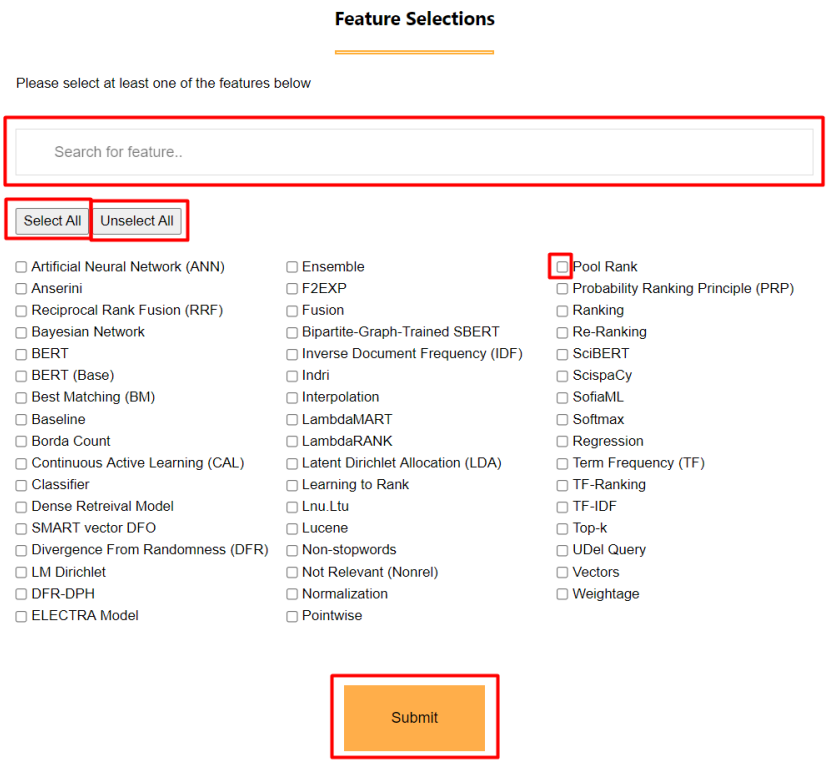


Figure 3. *Main features of the web application*

By referring to the main section of the web application, the components which are highlighted in red boxes such as the search bar, ‘Select All’ button, ‘Unselect All’ button, checkboxes and the ‘Submit’ button are the main functionalities of the front-end which allows user to interact with and handles user input consequently as shown in Figure 3 above. The functionalities of each component have already been tested during the testing phase to ensure they are working correctly as expected.

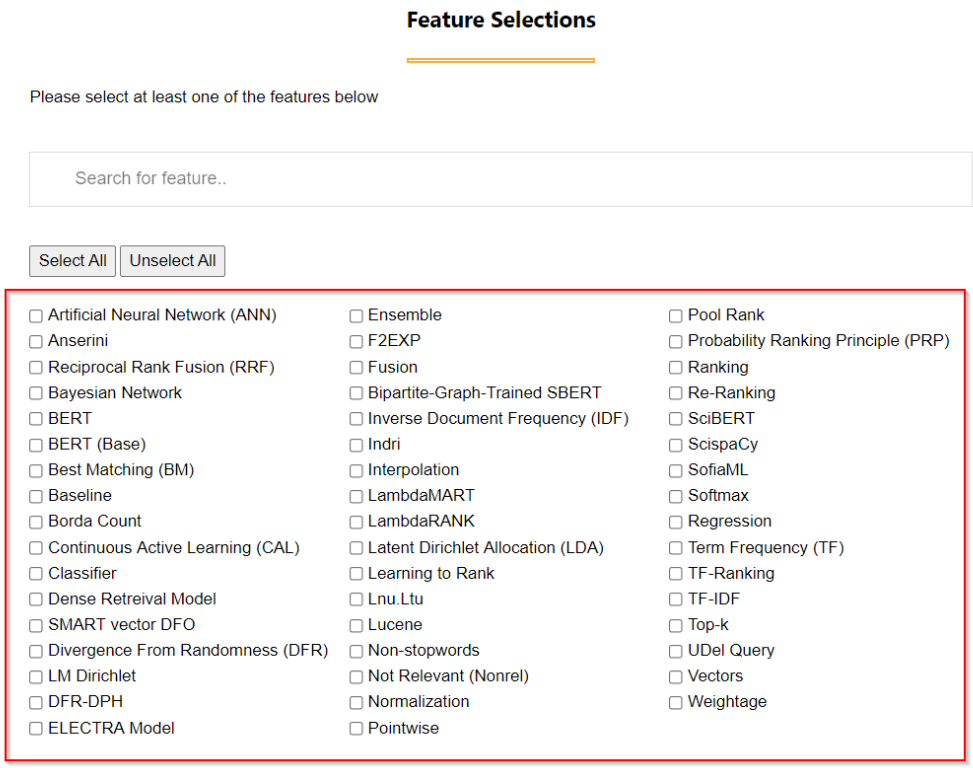


Figure 4. *Features available to choose from*

Based on Figure 4 above, users will be required to select at least one of the features from the choices that are available to choose from by clicking on the box provided next to the feature. The choices selected will then be encoded as ‘1’ as an input for the features selected and ‘0’ for features not being selected as the parameters required for our classification model are mainly categorical variables.

An example of how the front-end will be after the user has selected 4 features which are ‘Bayesian Network’, ‘Best Matching (BM)’, ‘Fusion’ and ‘Weightage’ can be found below in Figure 5.

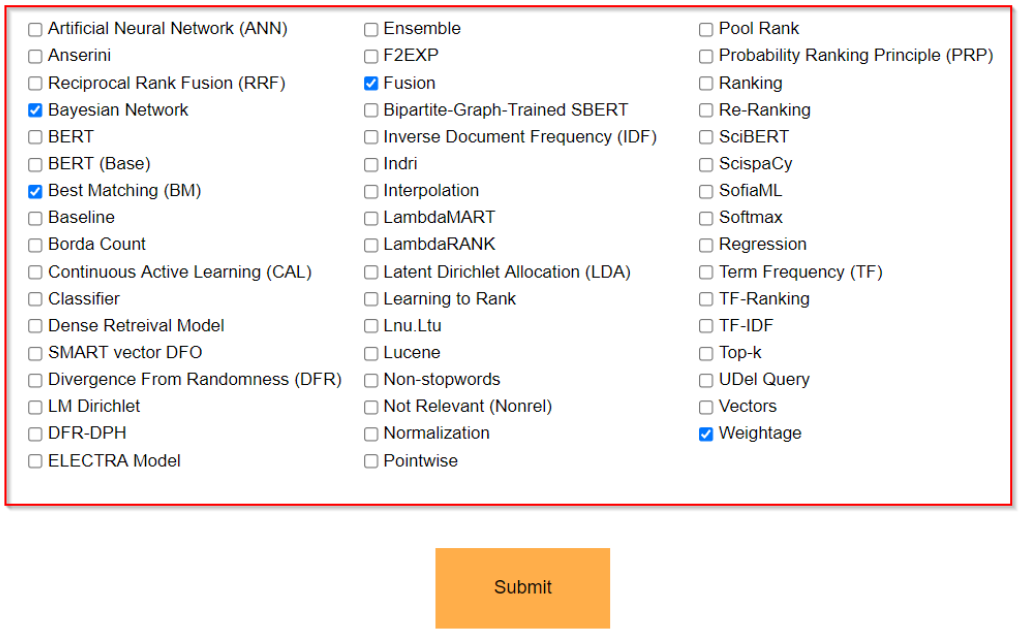


Figure 5. *Selecting 4 features 'Bayesian Network', 'Best Matching (BM)', 'Fusion' and 'Weightage'*

Next, after the user has submitted their choices by clicking on the 'Submit' button, our web application will display a total of 3 outputs which are the prediction results, followed by the model's confidence and a bar chart that allows user to visualize and identify which of the features selected are more strongly correlated to the performance of an IR as compared to other features that were selected previously.

An example of the output of our web application based on the features that were selected earlier can found below in Figure 6.

Your Model Performance:

High Performance

Your Model Confidence:

0.573

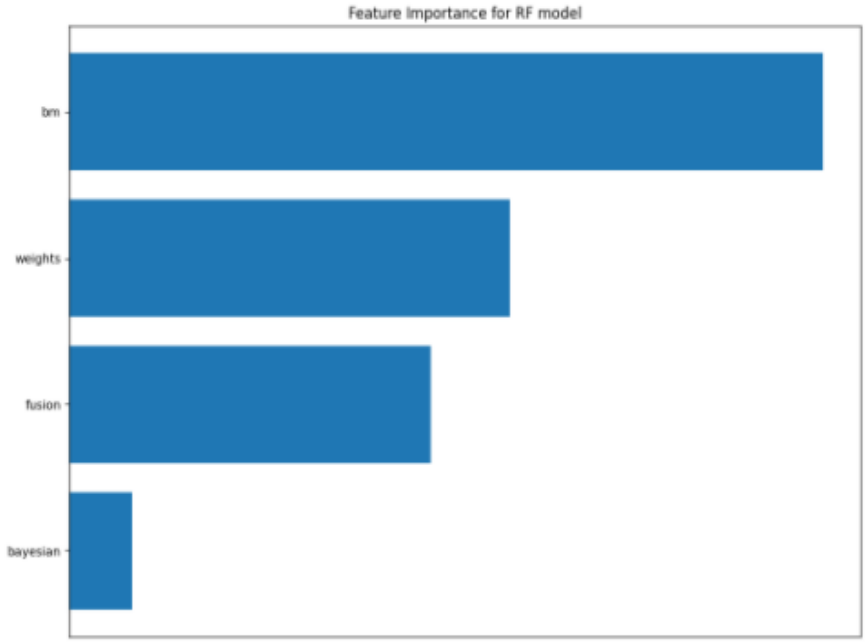


Figure 6. *Prediction results*

3.2 How are Requirements met

The table shown below are the functional and non-functional requirements that our product is required to meet :

Table 1. *Functional and non-functional requirements of our product*

Requirement ID	Category of requirements	Description of requirement
F1	Functional requirement	The web application should be using the best performing classification algorithm for the classification model obtained after evaluation
F2	Functional requirement	The web application should allow user to input data as an input parameters for the classification model to perform prediction
F3	Functional requirement	The web application should be able to output the prediction results of the classification model based on user input
F4	Functional requirement	The web application should be able to output a meaningful graph that is relevant for user to visualize the results obtained
F5	Functional requirement	The web application should be able to output the model's confidence in prediction
NF1	Non-functional requirement	Time taken to generate the results from the web application will not exceed 5 seconds
NF2	Non-functional requirement	The web application should be user-friendly
NF3	Non-functional requirement	The web application should have user input validation

The table below will show how each Requirement ID that were listed in Table 1 above has been met throughout our project :

Table 2. *Implementations to meet Functional and non-functional requirements of our product*

Requirement ID	How did we meet the requirements ?
F1	The best performing classification algorithm which is the Random Forest classifier, where the performance has also been improved with tuned parameters using a random search has been selected as our final classification model in our back-end.
F2	We have implemented the checkboxes in our front-end as shown earlier in Figure 4, which is the primary input data that we will be collecting from the user and what our classification model will be receiving as its input parameters to perform prediction once the user has clicked the 'Submit' button.
F3	The prediction results that will be displayed on our web application are either a 'High Performance' which has a green background as shown earlier in Figure 6, or a 'Low Performance' which has a red background that indicates that the performance of the IR system is low.
F4	The graph that our web application will display is a bar chart . This reason for selecting a bar chart as our main visualization is because it not only relevant to use in our project but it has also allowed users to visualize the ranking of the features which is meaningful for the user to identify which of the features that were selected by the user are more important that strongly correlates to the performance of an IR as compared to other features that were selected.
F5	The model's confidence in prediction will be displayed as the second output of our web application as shown in Figure 6 earlier. The range of the model's confidence will be ranging from 0 to 1, where the closer the value is to either 0 or 1, the more confident we are that our model prediction results are correct.
NF1	The response time which is the time taken to generate the prediction results on the web application based on the user's input once the user clicks the 'Submit' button has been tested during our testing phase. This has been done to ensure that our web application is responsive and fast to keep a user's attention and interest. Therefore, various test cases with a different number of features selected have been carried out to test the response time and we have found that the response time will not exceed 5 seconds regardless of the number of features being inputted by the user.
NF2	Remote usability testing has been carried out to identify if our web application is user-friendly. Through the testing phase, we have found that our web application is easy to use and user-friendly with a mean score of 4.6 out of 5 from the data we have collected from our participants that were involved. Therefore, our web application can be considered as user-friendly based on the mean score we have obtained from 15 participants that were involved in our usability testing.

NF3	The input validation has been tested during our testing phase to ensure that there will not be any input from the user that will affect the system of our web application and ensured that our classification model will be receiving the expected input from the user as parameters to perform prediction. Therefore, if the user has clicked the 'Submit' button without selecting any of the features, an error message will be displayed stating ‘Please select 1 or more.’

3.3 Justification of Decisions Made

We will be discussing some of the important decisions that our team has made throughout the project in the upcoming sessions in order according to the project timeline.

3.3.1 Choice of dataset

Firstly, one of the most crucial decisions that we have to make at the start of the project is deciding on the dataset that we will be using for our project. Given that there are no websites where we can easily download a dataset that contains all the required information we need in just a single click, we have decided to build our own dataset based on the information that has been compiled by NIST (National Institution of Standards and Technology) that contains the test collection for the outcome of all five rounds of TREC-COVID from the website as shown in Figure 7 below.

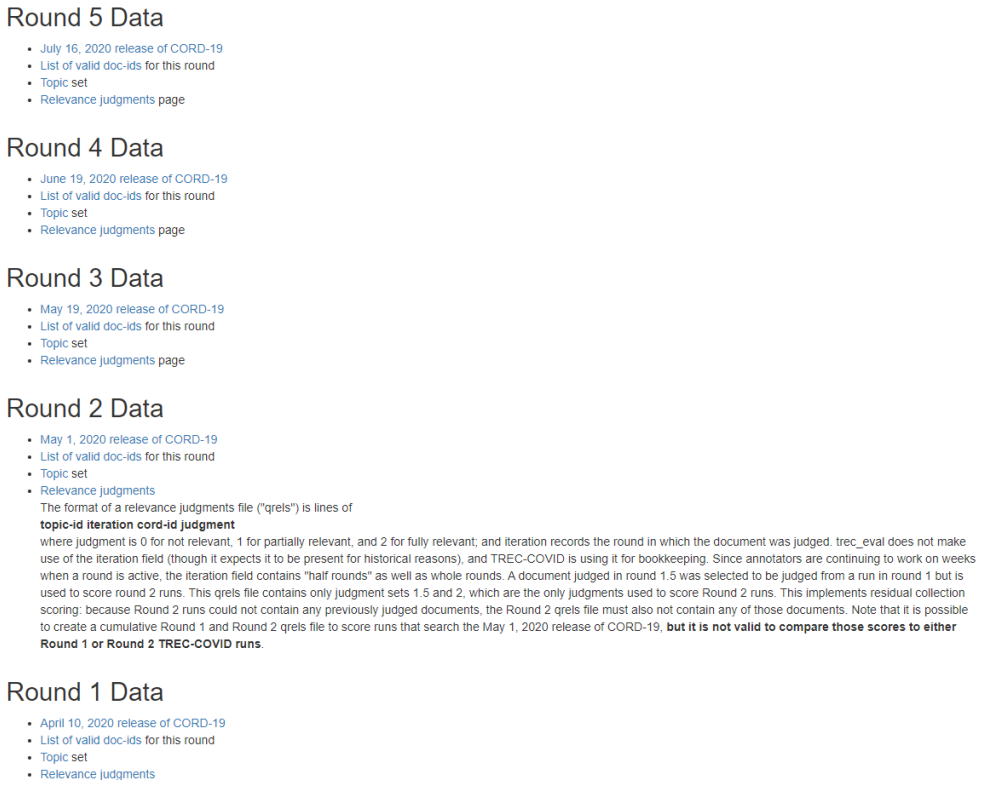


Figure 7. *TREC-Covid Archive*

The reason for choosing to build our own dataset from scratch from this website is due to its suitability and relevance of our project scope. For example, there will be a list of participating teams along with their scores for their IR system in each round of the TREC-COVID as shown below in Figure 8.

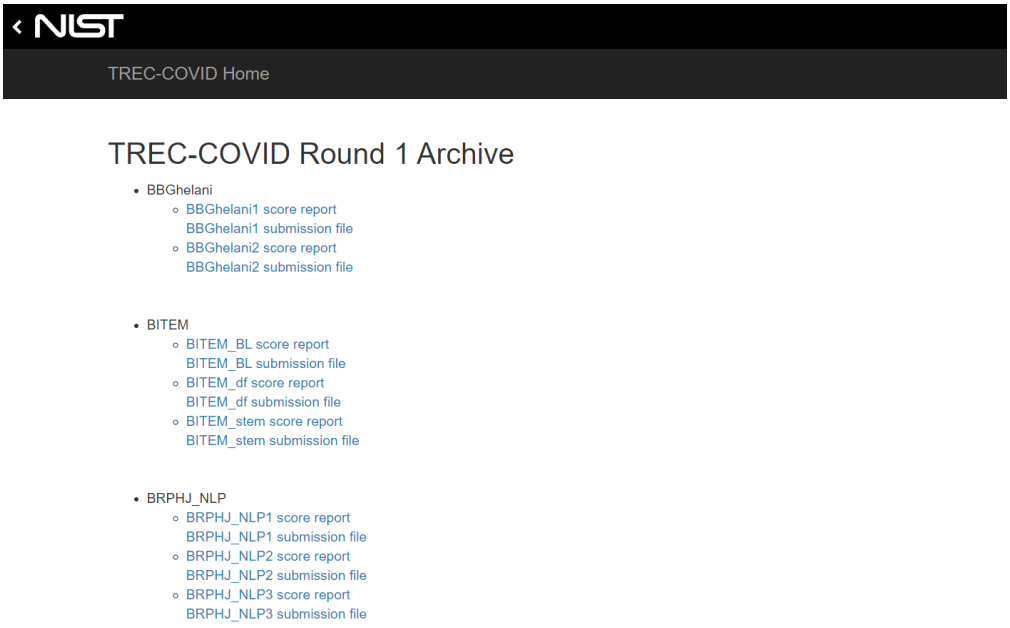
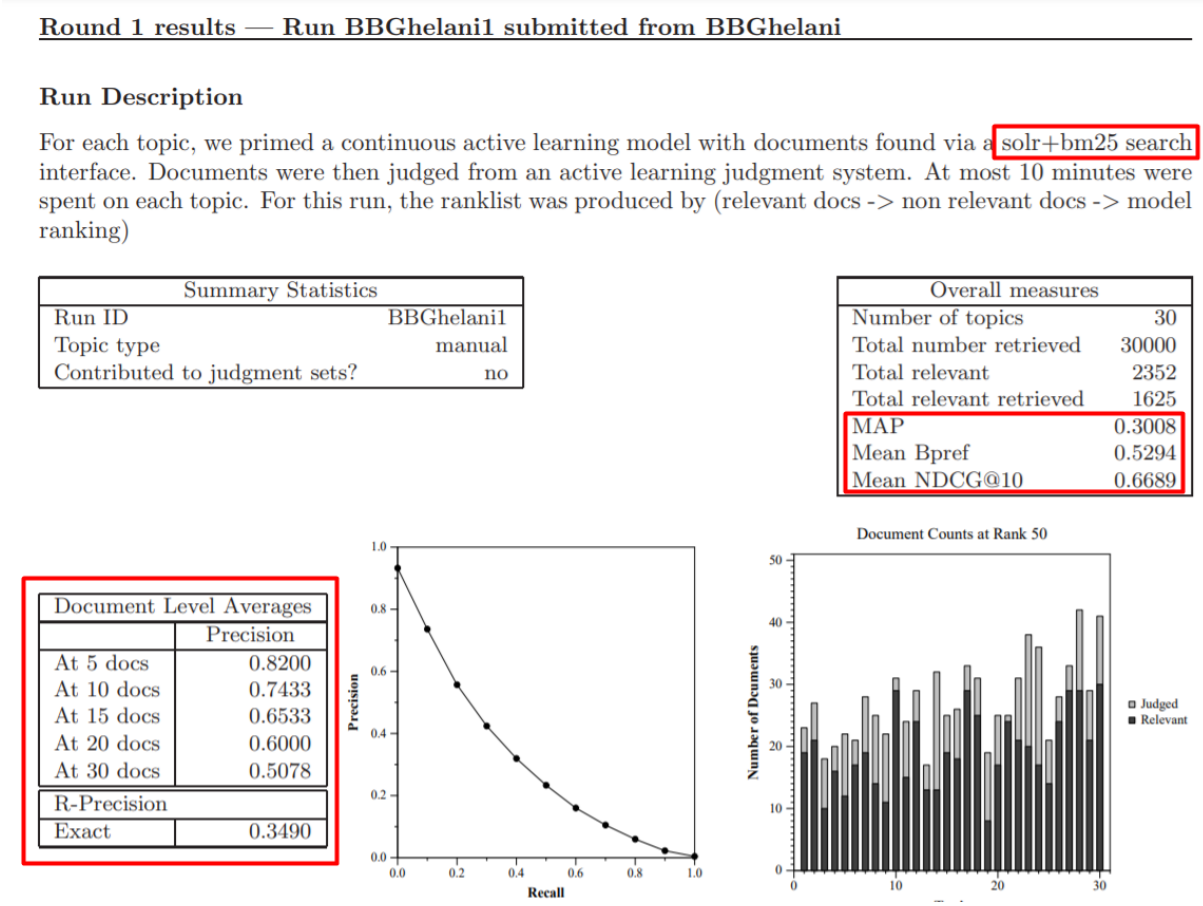


Figure 8. *TREC-Covid Archive with submissions*

From here, if we have clicked on an example of a run from a participating team ('BBGhelani'), it will lead us to a page in a pdf format where there will be information regarding methods or algorithms used and the scores of the evaluation metrics obtained from a participating team as shown below in Figure 9.



Round 1 results — Run BBGhelani1 submitted from BBGhelani

Run Description

For each topic, we primed a continuous active learning model with documents found via a solr+bm25 search interface. Documents were then judged from an active learning judgment system. At most 10 minutes were spent on each topic. For this run, the ranklist was produced by (relevant docs -> non relevant docs -> model ranking)

Summary Statistics	
Run ID	BBGhelani1
Topic type	manual
Contributed to judgment sets?	no

Overall measures	
Number of topics	30
Total number retrieved	30000
Total relevant	2352
Total relevant retrieved	1625
MAP	0.3008
Mean Bpref	0.5294
Mean NDCG@10	0.6689

Document Level Averages	
	Precision
At 5 docs	0.8200
At 10 docs	0.7433
At 15 docs	0.6533
At 20 docs	0.6000
At 30 docs	0.5078
R-Precision	
Exact	0.3490

Figure 9. *Example of a run in PDF format*

Therefore, our dataset is built by retrieving the methods or algorithms used and the scores obtained from each participating team which are involved in the TREC-COVID competition through text processing and manual data cleaning when necessary.

3.3.2 Creation of class label

To build a supervised machine learning algorithm for our classification model, a class label or a response variable which is the performance of an IR system (High/Low performing) in this case will be required for us to perform predictions. Given that there are no indications on whether the performance of the IR by each participating team is a high or a low performance as shown in Figure 9 earlier, we have decided to create our own class label by taking into consideration of the evaluation metrics obtained from each IR, as they provide us information on how these IR performs across each evaluation metrics.

Therefore, we have decided to make good use of these information to create our class label which is the performance of an IR system (High/Low performing) by calculating the average score values of the evaluation metrics as the scores from the evaluation metrics are values between 0 and 1, so there will not be any issues with heavily weighted average scores in this

case. Therefore, after the average score has been calculated, we will assign our class label to 1 (High performing) if it is greater than the average scores calculated, else 0 will be assigned (Low performing) for each run of a participating team, which is considered one of the important decisions we have made in our project.

3.3.3 Evaluation metrics used

Regarding evaluation metrics, although accuracy is known to be one of the most popular evaluation metrics that are most commonly used to evaluate the performance of a model, where we have initially thought of using it to evaluate the performance of our model as well.

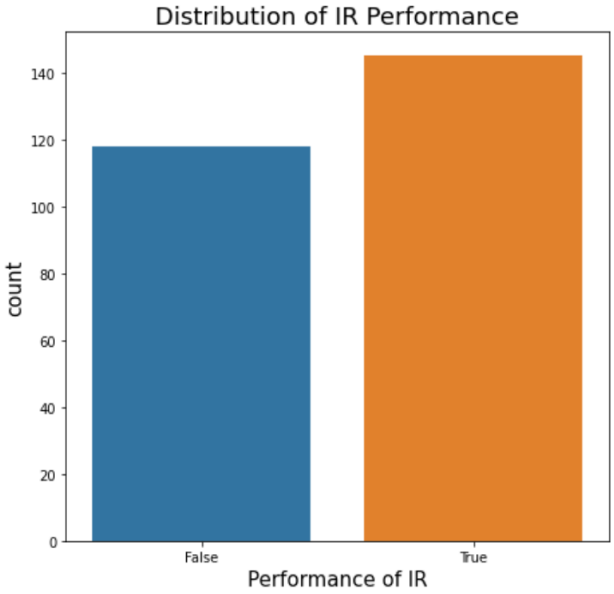


Figure 10. *Distribution of class label*

However, we have eventually decided not to use it due to the imbalanced nature of our dataset as shown in Figure 10 above, where the number of 'High performing'/'True' values for our class label is greater than the number of 'Low performing'/'False'. Therefore, it will be very likely that our model will predict more 'High performing' as compared to 'Low performing' IR systems.

Besides that, a high accuracy score obtained from our model does not necessarily mean that our classification model is performing well, as using accuracy as an evaluation metric is usually more ideal when the dataset is balanced where the number of 'High performing' and 'Low performing' class labels are equally distributed.

Hence, we have considered other alternatives such as F1-score, Log-loss and AUC value from an ROC curve which are more appropriate to evaluate the performance of our classification model for an imbalanced dataset. The performance of our model will be highlighted in the upcoming sections.

3.3.4 Choice of machine learning algorithm

We have decided to use the Random Forest algorithm for our classification model as it is a supervised learning algorithm which is suitable for us to use.

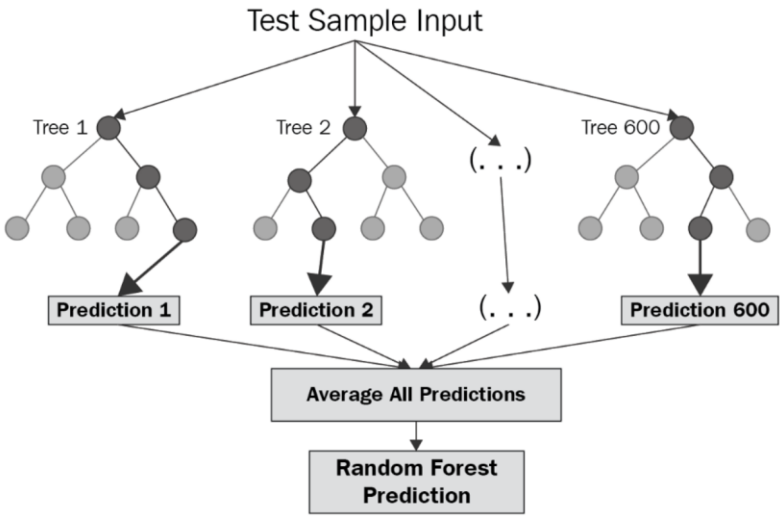


Figure 11. *An example of Random Forest classifier (Trehan, 2020)*

Besides that, the algorithm is known as an ensemble of decision trees which are trained using the 'bagging' approach that considers the results of a combination of learning models as shown in Figure 11 above, which increases the overall stability in prediction (Donges, 2021).

In addition, the relative importance of each feature on the prediction can also be measured easily using the Sklearn library in Python for a Random Forest algorithm. This has allowed us to see how much the tree nodes that use a specific feature reduce impurity across all trees in the random forest. This library will also compute the score of each feature automatically after training the model and scaling them up so that the sum of importance will be equal to one, which has provided us some extra information that not all algorithms will provide. Not to mention, one of the reasons why we have selected the Random Forest algorithm over other popular classification algorithms such as the Logistic regression model is because it tends to perform better in an imbalanced dataset as shown in Figure 10 earlier.

3.3.5 Hyperparameter tuning using Random Search

After evaluating the performance of our base Random Forest classification model, we have decided to perform hyperparameter tuning by tweaking the parameters and finding the best ones to further enhance the performance of our Random Forest classification model.

From here, we have to decide if we should perform hyperparameter tuning on our Random Forest model through grid search or random search which are both popular techniques that are

commonly used. Initially, although it may seem that grid search will be the best choice, as every possible hyperparameter combination is tested to find the optimal hyperparameter values (Pferiffer, 2019). However, as the number of dimensionality or parameters increases, the number of hyperparameter combinations to search for will also increase, as each additional parameter would also increase the number of evaluations exponentially where more time and computational resources will be required to find the optimal hyperparameters.

Therefore, given that there are a total of 48 features including the class label in our current dataset, we have decided to use a random search method which will be more practical for us to perform hyperparameter tuning on our Random Forest classification model as compared to the grid search method. Random search is a method where it randomly selects combinations of hyperparameters to train a model, which is typically faster as the number of combinations to attempt can be controlled, which may lead to more efficient computational power used as compared to grid search (Pferiffer, 2019). Furthermore, after identifying the best random hyperparameter combinations using Random Search, we have trained our Random Forest classifier using the best parameters we have found earlier and evaluate the performance of our classification model using the evaluation metrics that we have used for our base model and make a comparison.

Performance of Base Random Forest model :

```
Model F1-score with Base Random Forest : 0.73333
Model Log-loss score with Base Random Forest : 0.51016
Model Precision score with Base Random Forest : 0.73333
```

Figure 12. *Performance of base Random Forest model*

Performance of Random Forest model after hyperparameter tuning :

```
Model F1-score with Tuned Random Forest : 0.75410
Model Log-loss score with Tuned Random Forest : 0.47330
Model Precision score with Tuned Random Forest : 0.74194
```

Figure 13. *Performance of Tuned Random Forest model*

Based on Figure 13, we can see that the performance of our Random Forest model has been improved after tuning our parameters using a random search method as compared to the results that we have obtained from our base model without any hyperparameter tuning in Figure 12 earlier. For example, we can see an increase in our F1-score from 73.33% to 75.41%, a decrease in our Log-loss score from 0.51 to 0.47 and an increase in our Precision score from 73.33% to 74.19%.

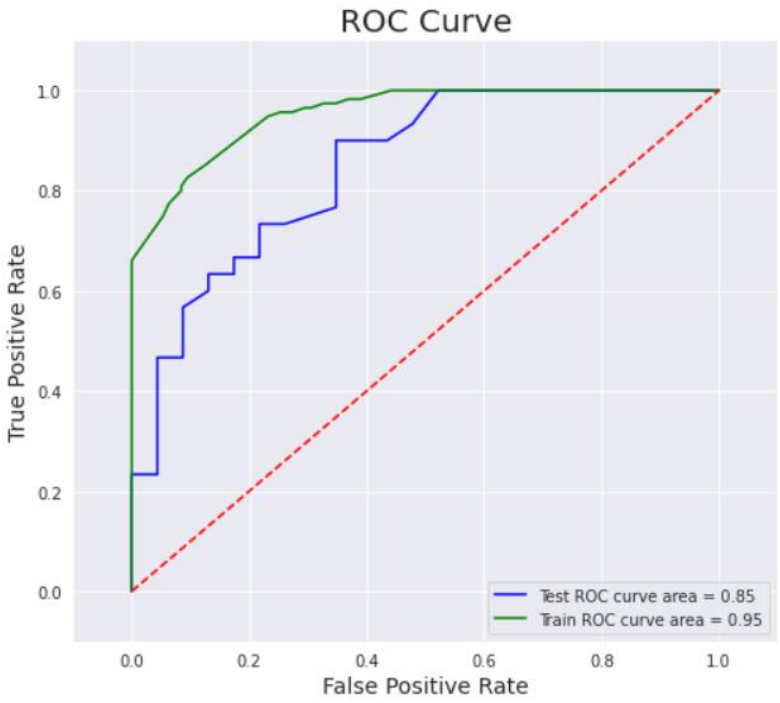


Figure 14. *ROC curve*

In addition, we have also plotted the ROC curve for our training and testing data to observe the AUC value of our classification model. Based on Figure 14, we can see that the AUC value obtained from our training model is 0.95 which is very high as compared to the AUC value obtained from the testing data which is 0.85, which could be a sign of overfitting in this case. Nonetheless, the AUC value is still considered to be good for our classification model in prediction. Therefore, the Random Forest classifier with tuned hyperparameters using a random search will be selected as our final classification model.

3.4 Limitations of Project Outcomes and Possible Improvements

3.4.1 Overfitting of classification model

Firstly, one of the limitations of the project outcomes is that our classification model is showing signs of overfitting where the performance of the model using training dataset is greater than the performance using a testing dataset as shown in the ROC curve in Figure 14 earlier in the previous section. Overfitting is known to be a common problem in the field of machine learning and data science where the model has learned the noise in the training data and made predictions based on it, which resulted in not being able to predict well on unseen data (Sagar, 2019). Hence, there are a few popular techniques in handling overfitting issues that we can consider to improve the model in generalizing well to new data.

One of the options we can consider to reduce overfitting is by including more data to train our model, as the current dataset that we have may not exhibit all the properties in the test data,

which has led to our model not being able to generalize well. However, we are unable to obtain a larger dataset aside from the ones which we are currently using from the TREC-COVID which consists of data from 5 rounds, which are the only relevant data to our project. Hence, we might only be able to enlarge our dataset by waiting for the results from the next rounds being released from TREC-COVID. Therefore, we might have to consider data augmentation, which is an alternative that enables the available dataset to appear slightly different and diverse each time it is being processed by the model to overcome overfitting.

Furthermore, given that we have many features with a limited amount of training data, we can try to perform other feature selection methods to only select the most important features for our training data, so that our model will not learn so many features that will eventually lead to overfitting.

In addition, overfitting could also be caused by the complexity of the Random Forest algorithm that we are currently using as our classification algorithm. A model with a higher complexity might have eventually learned the noise in our training data set and caused overfitting (Luis Amoros, 2021). Therefore, we can try to tweak our random forest model parameters to reduce the complexity of our model and reduce the tendency of our model to overfit.

3.4.2 Participants involved in usability testing

Furthermore, one of the limitations we have encountered in our usability testing is that the mean score obtained from participants regarding user-friendliness of our web application might be subjective as everyone may have their own opinions towards the user-friendliness of our web application. Besides that, the total number of participants involved in our usability testing is only a total of 15 participants.

Therefore, we believe that by reaching out to more people, we may be able to obtain a more accurate mean score that is more reliable. Not to mention, we are also able to obtain more additional feedback on things that we can implement to improve the web design of our web application and make further improvements on the user-friendliness of our website by involving a greater number of participants in our usability testing.

3.4.3 Design of web application

In addition, we have also evaluated the design of our web application through usability testing and found that there are still room for improvements that can be made on areas such as increasing the size of the checkboxes and increasing the font size for the feature names to enhance readability based on the feedback we have received from our participants. In fact, we have thought of a few ideas on features we can implement to improve the design and the user-friendliness of our web application. However, given that we are fairly new to web development, most of the time has been used to learn the fundamentals of web development from scratch. As for now, our main intention is to implement the functionality of the things we intended to do on our web application that meet the requirements. Therefore, we are unable to implement those

ideas that we have in mind due to the limited amount of time and resources available. However, we believe that with more time and experience we have in web development, we will definitely be able to improve the design of our web application by including more interesting features that could improve the overall experience of our user on our web application.

** Section 4 will be on the following page*

4 Methodology

4.1 System Design

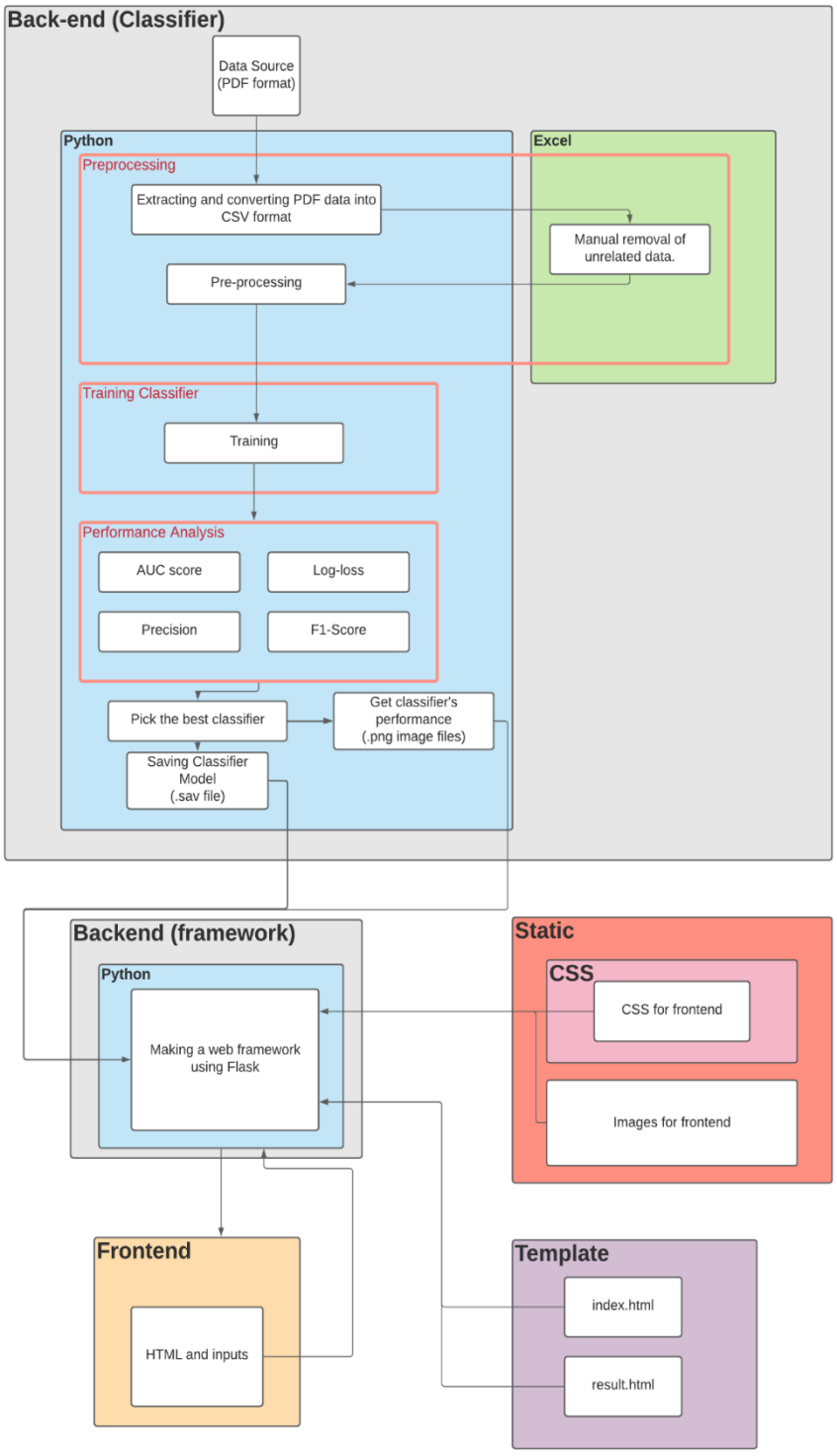


Figure 15. *System Design (overall architecture)*

Based on Figure 15 above, the final product consists of 2 parts which are the backend and frontend respectively.

Regarding the frontend, it has been implemented through the usage of HTML script and a CSS template. Besides that, most of the frameworks that have enabled the connection between the backend and the frontend are done through the usage of the Flask library in Python.

Furthermore, after receiving input from end users through the front-end, it will then be processed by Flask and then imputed to the classifier itself. Next, the output generated from the classifier will then be sent to the frontend by the Flask library as well.

Moreover, Flask will require both files which are “Static” and “Template” in order for Flask to load the required resources for the HTML. This is because Flask will retrieve any HTML files from the file “Template” and will retrieve any static image, CSS and JS from the “Static” file. Besides, code execution will be done using Python and any additional information such as the relevant images which are related to the performance of the classifier that are generated in Python can be displayed on a web page with the use of Flask.

The frontend will display any default pathway that is coded in the backend to be displayed to the website. For example, in the Sample Code 1 in the appendix section, the backend with the path of “/” (which is the default pathway) will run index() and display “index.html” and the path of “/predict” will run the predict() from the backend return a result. Next, results will then be displayed in “result.html” as it will contain the right HTML of how the result should be displayed on the frontend.

4.2 Data Preprocessing

Furthermore, regarding backend, the dataset used within the project is sourced from TREC-COVID 'Archive' from the NIST's website as mentioned earlier in Section 4.3.1 where all 5 rounds of data are combined and then processed by converting data from PDF to CSV format and applying the bag-of-words technique to extract information from the summary such as the algorithms or methods which are used by a specific IR system. Next, the data processed will be manually inspected using Excel to ensure that data is clean before being used for training.

The following figure will show the process of the data cleaning process for data quality assurance :

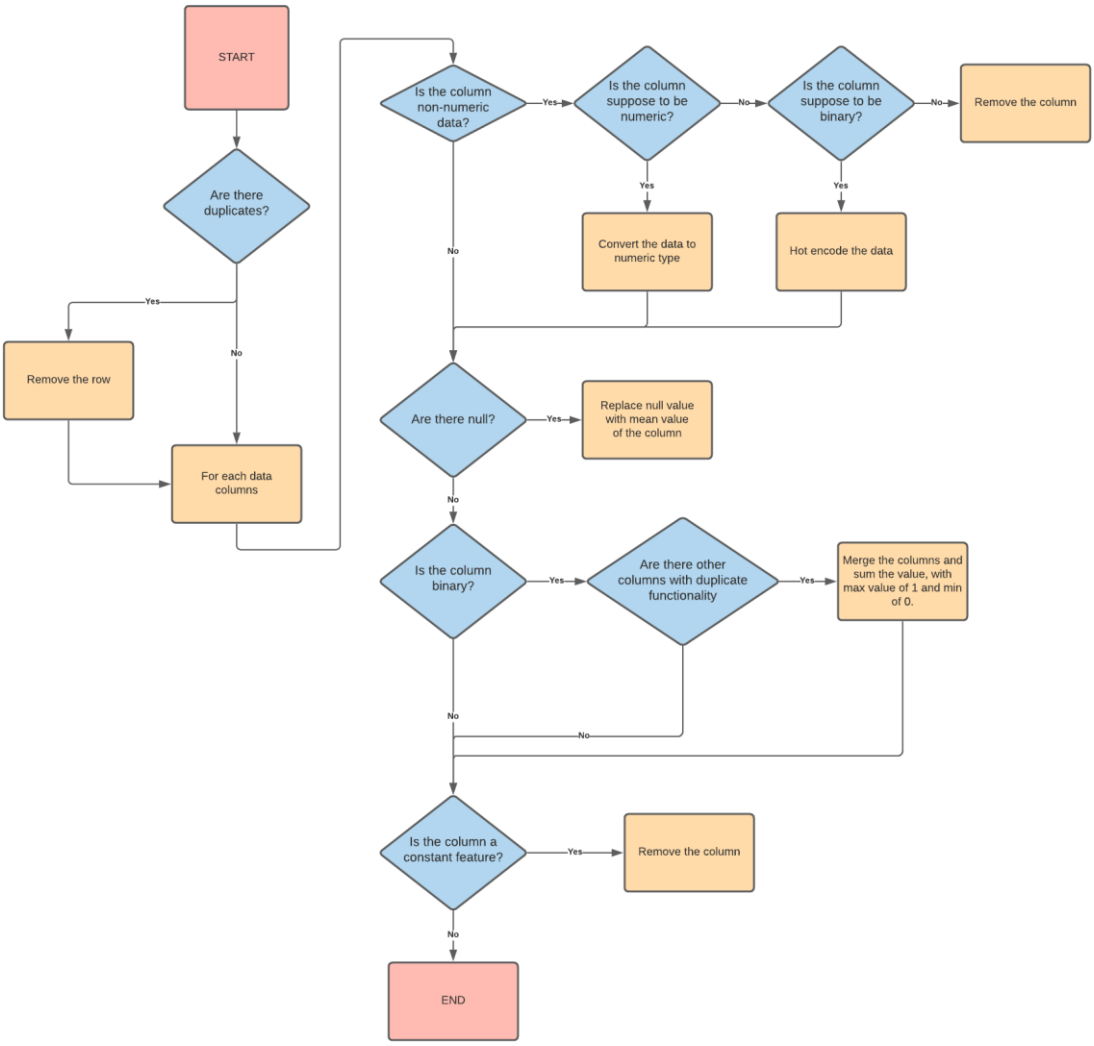


Figure 16. *Flow chart for data cleaning*

After most of the data cleaning process has been completed, the class label will be created by averaging the value of the performance metrics obtained from each IR system for each row in

the dataset, where the label column will have its current value being replaced with a one-hot binary format, with 1 and 0 indicated by whether the class label is lower or higher than the mean value of the class label column which represents the performance of the IR system in other words as mentioned earlier in Section 4.3.2.

4.3 Data Modelling

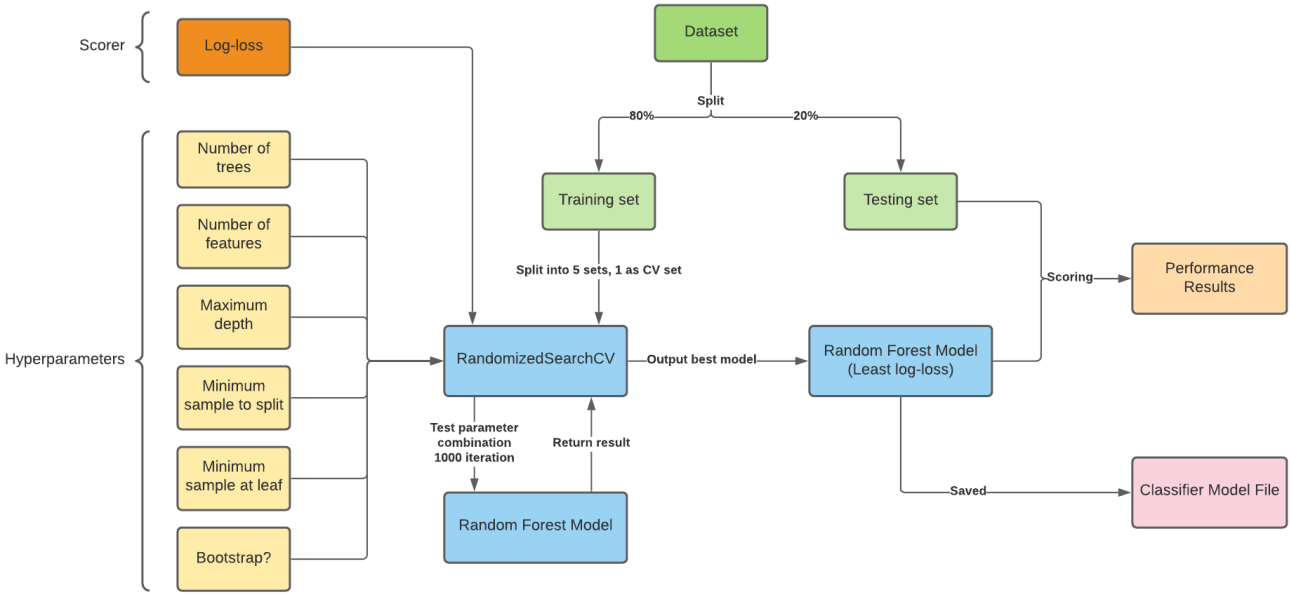


Figure 17. *Flow chart for data modelling*

The figure above represents the overall flow of the data modelling process we took to build our Random Forest classifier. Firstly, we have split our dataset into 80% training data and 20% testing data and train our base model without any hyperparameter tuning and observe the performance of our classifier through the evaluation metrics. Next, we have performed hyperparameter tuning which consists of a set of values for each of the hyperparameters to perform a randomized search using the Random Search method.

The Randomized search cross validator will then try out each combination from the hyperparameters and train the Random Forest with those hyperparameters with the goal of finding the set of hyperparameters that maximize the scorer, which is the lowest log-loss in our case. Therefore, the hyperparameters that would lead to the lowest log-loss score will then be used to build our new Random Forest model which was selected as our final classifier after evaluating the performance of the model through evaluation metrics once again.

5 Software Deliverable

5.1 Summary of software deliverable

5.1.1 What is delivered

A web application will be delivered as our final product which consists of a total of 3 main sections where the first main section of our product has already been covered in Section 4.1 earlier. Therefore, the main functionality of our web application is to predict the performance of an IR system based on the features that were selected by the user that will be sent as an input parameters to the classification model in our back-end, and the 3 outputs mentioned earlier are computed by the classification model in our back-end. Next, the outputs will be sent back to the front-end which is our web application where the results generated will not exceed 5 seconds.

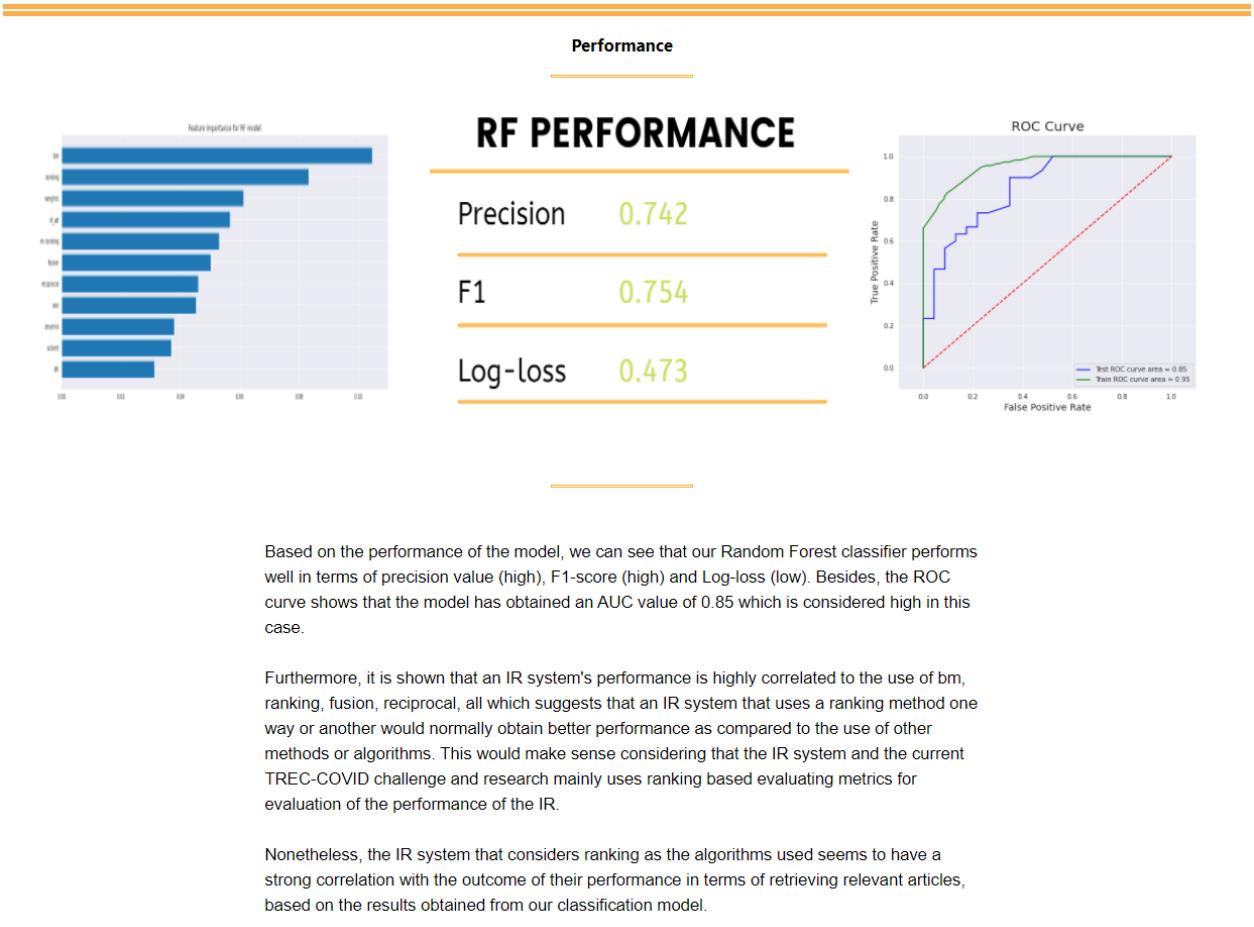


Figure 18. *A preview of the website (Middle)*

Furthermore, under the second section of our product, we will be providing some information regarding performance of our Random Forest classifier for users as shown above in Figure 18.

From the top left, we have included the top 10 features which are the methodology and algorithms that strongly correlates to the performance of an IR, followed by the evaluation metrics score obtained from our Random Forest classifier and lastly the ROC curve where users can identify the obtained AUC value from the training and testing data. Next, the paragraphs below are a descriptive summary for each of the static items which were highlighted earlier.

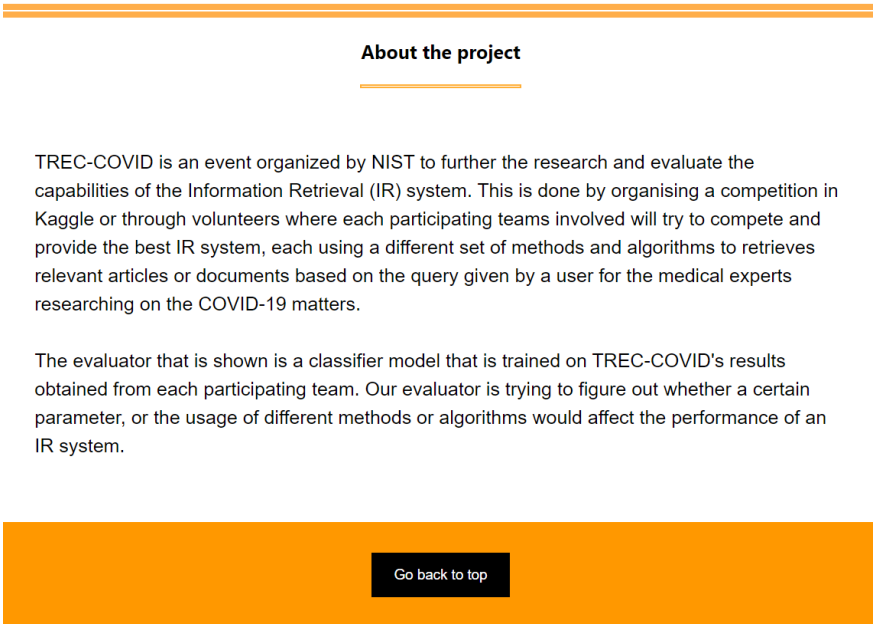


Figure 19. *A preview of the website (Bottom)*

For the last section of our web application, information regarding data that we have used to build our classification model and the purpose of our project or classification model has also been provided under this section.

5.1.2 Sample screenshots of usage

Given that we have already provided an example of what the output will be like if the user has selected a few features that have resulted in a 'High performing' prediction as shown in Section 4.1 earlier. We will be providing two more additional examples which would result in a 'Low performing' prediction and the result that will be shown if the 'Select all' button feature has been used to select all features and perform a prediction.

Sample that would result in 'Low performing' :

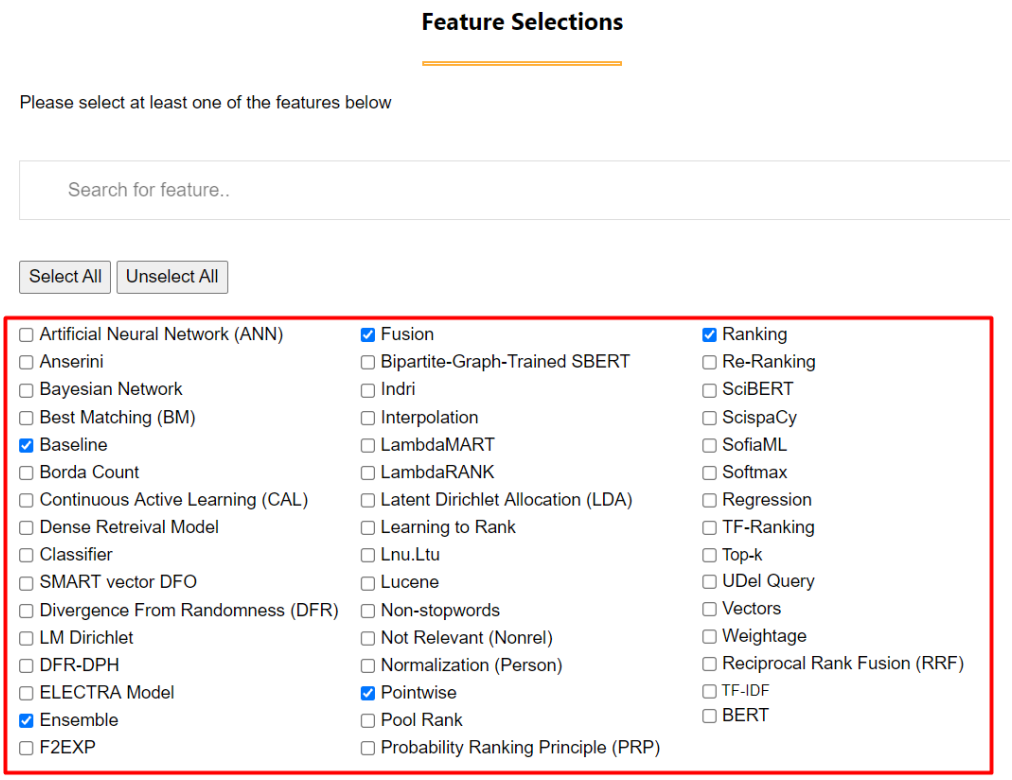


Figure 20. *Selected features that would result in 'Low Performance'*

Your Model Performance:

Low Performance

Your Model Confidence:

0.322

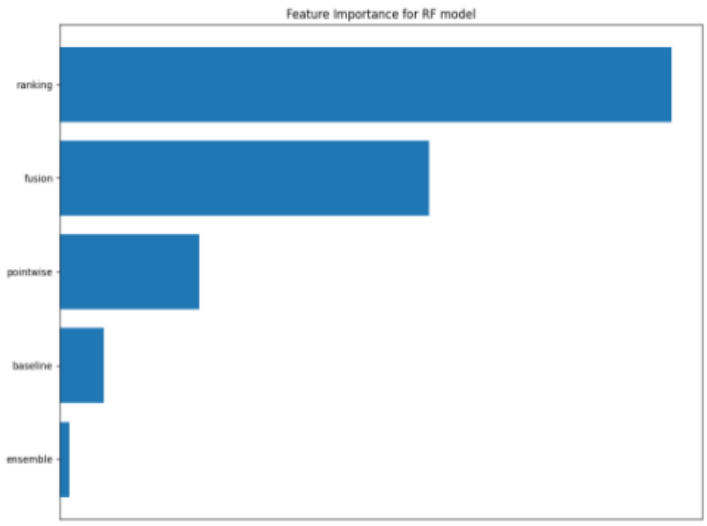


Figure 21. *Prediction result*

Sample if all features has been selected :

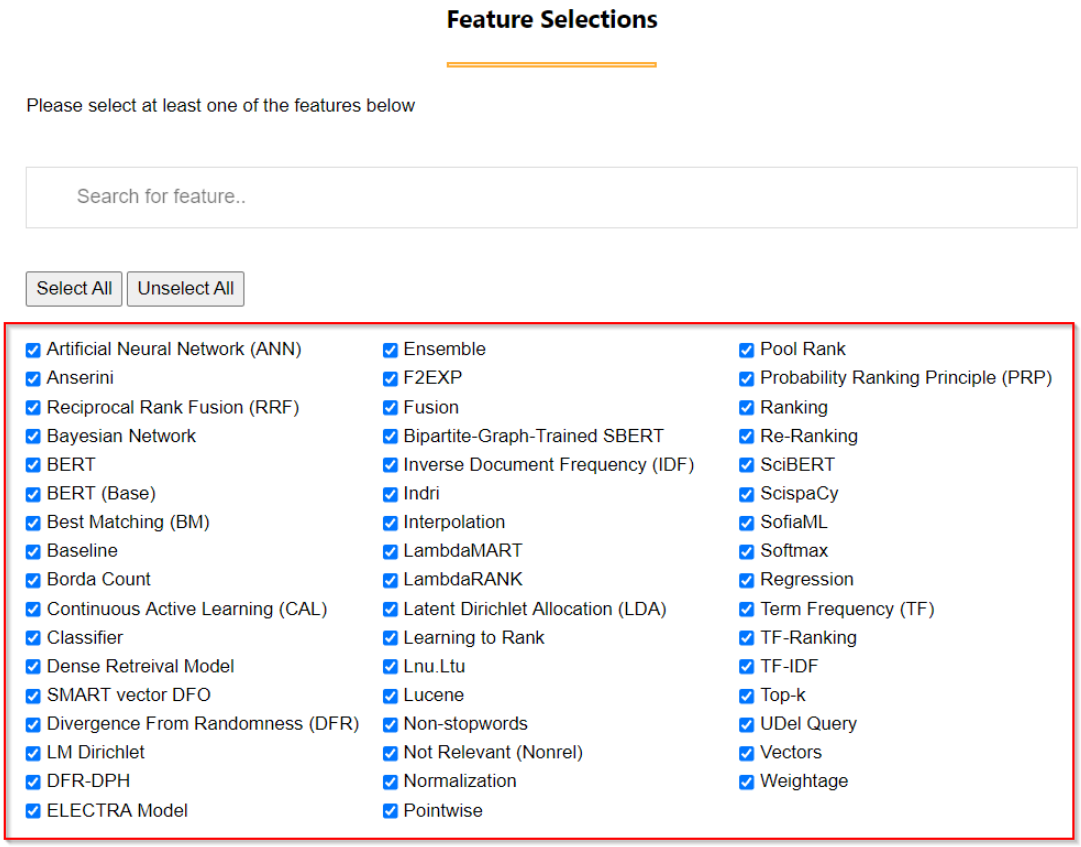


Figure 22. *After selecting all features through 'Select All' button*

Your Model Performance:

High Performance

Your Model Confidence:

0.584

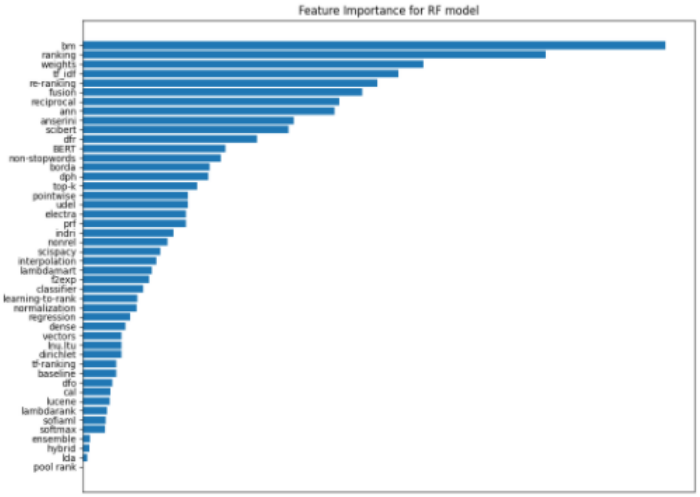


Figure 23. *Prediction results*

5.2 Summary and discussion of software qualities

5.2.1 Robustness

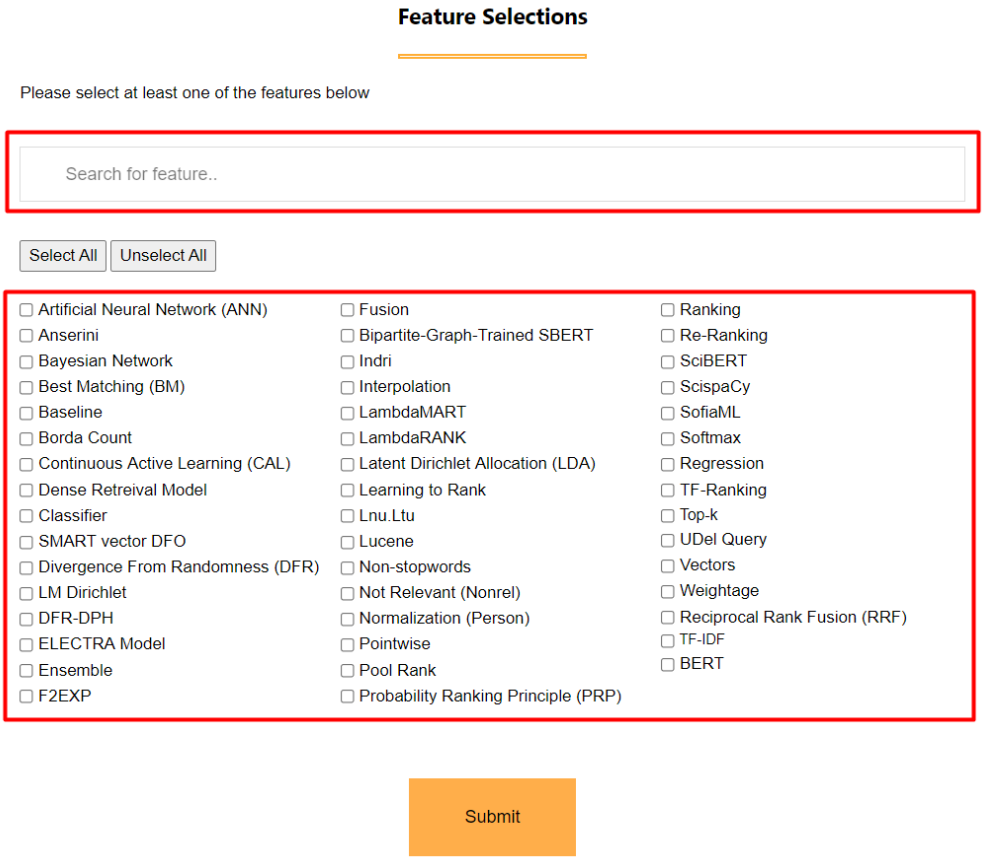


Figure 24. *'Search bar' and checkboxes of web application*

In terms of robustness, users are limited to what they can interact with in our web application as they are only able to interact with the 'Search bar' and the checkboxes as shown in Figure 24 above. The input from the 'Search bar' will not affect the back end, as it will only be used by the user to search the features that they would like to find in the front-end. Therefore, we will only be validating the checkboxes and ensuring that all input data are handled properly where there will not be any unanimous input parameters that will be sent to our classification algorithm in the back end. For example, an error message will be shown stating 'Please select 1 or more.' in red text if the user has clicked the 'Submit' button without selecting any features as shown below in Figure 25.

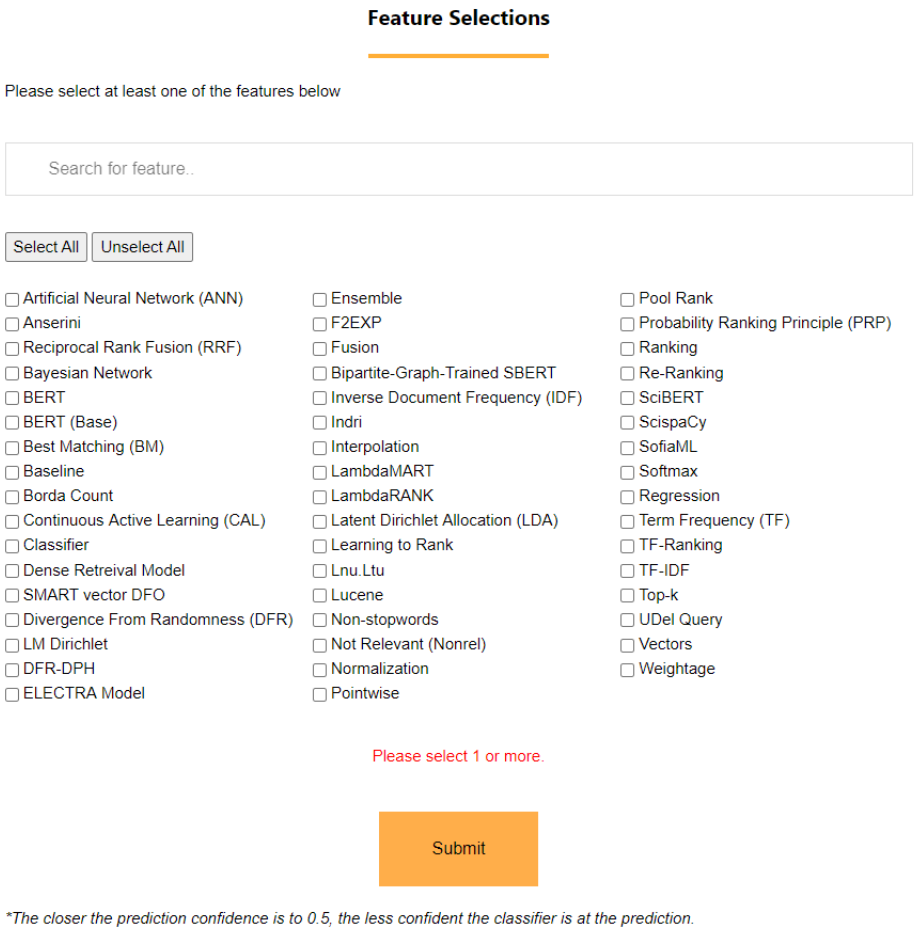


Figure 25. *Error message from invalid input*

5.2.2 Security

Given that we do not have any private and confidential information that we have in our dataset, we did not perform any security testing in our testing phase where the main purpose of it is to uncover the potential vulnerabilities of the system and ensure that the important data is protected from possible intruders. Hence, we do not see a need for performing any security testing to prevent our software from being attacked by hackers.

Besides, the only user inputs are obtained from the 'Search bar' and the checkboxes with strict constraints on what the user is allowed to do where the inputs are only constrained to binary inputs. In addition, our repository for front-end development has also been privatized, which will not allow any third parties in accessing our codes and modifying it.

5.2.3 Usability

In terms of usability, although we can see that there are still some improvements we can make in our web application as mentioned earlier Section 4.4.3, we have still managed to obtain a mean score of 4.6 out of 5 from the data we have collected from our participants that were involved in our remote usability testing. Therefore, our web application can be considered as user-friendly based on the mean score we have obtained from 15 participants that were involved in our usability testing. As the design of our web application is simple yet fully functional which has ensured that each of the product requirements are met.

5.2.4 Scalability

Given that our project is a small-scale project, we have not considered the scalability aspect of our project and the need to conduct any scalability testing which is a non-functional testing method which is most commonly done to identify how well the system performs when there is a projected increase in user traffic. However, we will consider scaling our project to deliver the same response time for different levels of user load if we have decided to continue working on our project in the future.

5.2.5 Documentation and Maintainability

In terms of documentation and maintainability, we have ensured that the important functions and codes are documented properly with sufficient information in each section, which will allow any other programmer who intends to continue our work that we have done to quickly understand the code easily and then make the necessary changes accordingly. Besides that, the function documentation also contains a description of what the function does and output that will be returned from the function. In addition, consistent code indentation has also been applied in our function which makes the code more organized and easier to read and follow along.

5.3 Sample Source code

Code that is used to start the backend of the server can been found in Sample Code 1 and Sample Code 2 in the Appendix section.

Code that is used to run the webpage of the prediction model can been found in Sample Code 3 in the Appendix section.

6 Critical Discussion

6.1 Deviations from initial project proposal

6.1.1 Hosting web application online

Initially, we have only planned to host our web application locally as what we have proposed earlier in our initial project proposal. However, after having some discussion about the benefits of hosting our web application online as compared to hosting it locally, we have eventually considered to host our web application online using PythonAnywhere which is a web hosting service that is suitable for us, as it uses a Python programming language on its platform.

Besides that, given that we are required to conduct a usability test to ensure that we are able to evaluate the user-friendliness of our web application which is one of the non-functional requirements we are required to meet. However, as we are not allowed to meet up face-to-face in the campus to perform a usability test in-person at the moment.

Therefore, one of the primary advantages of hosting it online is that we are able to conduct our remote usability test easily through Zoom meeting which is a screen-and audio-sharing tool that is widely accessible among our participants, by sharing the link of our web application that is hosted online to participants. In addition, one of the key benefits of remote usability testing is that it is generally faster and more cost-efficient as there would not be any need to travel to conduct the test (Carr, 2020). Not to mention, given that our web application is hosted online, this has also allowed us to test a diverse geography of participants from other areas with minimal effort through remote usability testing (Morales, 2020). Hence, we are glad that we are able to execute the change in hosting our web application online that was not initially included in our project proposal. Nonetheless, aside from this addition that we have included, our overall project has been executed according to the overall architecture of our project design as shown in Figure 15 of section earlier.

6.1.2 Project scheduling

In terms of project scheduling, we have been using our Gantt chart as our project schedule to get an overview on the project timeline for each project deliverable. This will ensure that we are able to keep track of our progress on each task easily and ensure that each task will be completed within the given timeframe which were scheduled earlier. However, as we are previously unaware of the due dates that will be set in this semester for the project deliverables, we are required to readjust the project timeline accordingly to the due date that has been released during this semester. Therefore, we have to reevaluate our priorities in completing certain deliverables before proceeding with the next one and so on. The changes made in our project scheduling is crucial for us to ensure that we are able to manage our time and complete a project deliverable within a specific timeframe that eventually leads to the project's overall

goals. Nonetheless, we are still able to complete all the project deliverables within the due date given, after some careful planning on the time that will be allocated for each project deliverables.

7 Conclusion

In summary, the final product that we have developed is a web application that has been implemented with the core features which has allowed users to predict the performance of an IR system based on a set of features that will be selected by the user effectively through the classification model that we have implemented using a Random Forest algorithm.

Furthermore, we have also found that the usage of methodology or algorithms such as ‘‘Best matching (BM)’, ‘Ranking’ and ‘Weighting’ within an IR system tends to strongly correlate to the performance of an IR system as compared to other methodology or algorithms that were used in an IR system through our classification model that we have built.

Besides that, although there are still some improvements that can be done to reduce the overfitting in our prediction model and to improve the design of our web application. Nonetheless, our final product has still successfully managed to meet all the functional and non-functional requirements within the project timeline despite the pandemic.

All in all, our final product has not only provided some information regarding features that corresponds to the performance of an information retrieval system, but it has also provided more opportunities for future work regarding TREC-COVID to improve the quality of the articles being retrieved in the future.

8 Appendix

8.1 Sample Code 1

```python
@app.route("/")
def index():
    """
    Default function to run if running index.html page.

    Return:
        Return index.html on url of /.

    """
    return render_template("index.html")



@app.route("/predict", methods = ["POST","GET"])
def predict():
    """
    Predict function, runs when user access /predict of html.

    Return:
        Return result.html on url of /predict.

    """

    #if it's POST request
    if request.method == 'POST':
        lst = getChecks()

        #check if list has 1's
        if 1 not in lst:
            #if no 1's found, return to default page with message.
            return render_template('index.html', result = "Please select 1 or more." , txt_color = "red")

        #compile result : [model result, model probability, result's graph]
        res = [model.predict([lst])[0],model.predict_proba([lst])[0][1], get_graph(lst)]

        color = ""
        res_txt = ""

        #change colour of result background
        if res[0] == True:
            #if model result is true, return green
            color = "4FE34F"
            res_txt = "High Performance"
        else:
            #else red
            color = "FF7F7F"
            res_txt = "Low Performance"

        #return all the compiled result to result.html and render it out on url of /predict
        return render_template('result.html', result = res_txt, color = color, confidence = round(res[1],3), graph = res
    else:
        #if it's GET request, return default page
        return render_template('index.html')


def get_graph(input_list):
    """
    Get graph of the predict result's feature importances of input_list
```

Figure 26. *Sample code 1 of backend server code that shows index() function and predict() function in flask_app.py*

8.2 Sample Code 2

```
def get_graph(input_list):
    """
    Get graph of the predict result's feature importances of input_list

    Argument:
        input_list : a list of 1's and 0's to indicate which feature was selected.


    Return:
        Return graph of the feature importances of the given input_list

    """

    columns = ['ann', 'anserini', 'bayesian', 'bm', 'baseline', 'borda', 'cal',
       'classifier', 'dense', 'dfo', 'dfr', 'dirichlet', 'dph', 'electra',
       'ensemble', 'f2exp', 'fusion', 'hybrid', 'indri', 'interpolation',
       'lambdamart', 'lambdarank', 'lda', 'learning-to-rank', 'lnu.ltu',
       'lucene', 'non-stopwords', 'nonrel', 'normalization', 'pointwise',
       'pool rank', 'prf', 'ranking', 're-ranking', 'scibert', 'scispacy',
       'sofiaml', 'softmax', 'regression', 'tf-ranking', 'top-k', 'udel',
       'vectors', 'weights', 'reciprocal', 'tf_idf', 'BERT']
    feature_importance = model.best_estimator_.feature_importances_

    filtered_col = []
    filtered_imp = []

    #get selected feature and put them into a list
    for i in range(len(columns)):
        if input_list[i] == 1:
            filtered_col.append(columns[i])
            filtered_imp.append(feature_importance[i])


    filtered_col = np.array(filtered_col)
    filtered_imp = np.array(filtered_imp)

    sorted_idx = filtered_imp.argsort()
    top_columns = filtered_col[sorted_idx]
    top_columns_score = filtered_imp[sorted_idx]

    #create graph
    fig = Figure()
    axis = fig.add_subplot(1, 1, 1)
    axis.barh(top_columns, top_columns_score)
    axis.tick_params(
        axis='x',
        which='both',
        bottom=False,
        top=False,
        labelbottom=False)
    axis.set_title("Feature Importance for RF model")

    fig.set_size_inches(12, 9)

    #change graph into format of which the HTML can read
    img = io.BytesIO()

    fig.savefig(img, format="png")
    img.seek(0)
```

Figure 27. *Sample code 2 of backend server code that shows get_graph function in flask_app.py*

8.3 Sample Code 3

```
}

ul {
    columns: 3;
    -webkit-columns: 3;
    -moz-columns: 3;
}

</style>

<!-- Title -->
<center>
<div style="background-color:rgb(255, 174, 74);" id="totop"><big>
<br>
<br>
<br>
<h1 style="font-size:55px;"><b>Prediction in performance of Information retrieval system </b></h1>
<br>
<hr>
</big>
</div></center>
<!-- End Title-->

<!-- Start Border 1 -->

<center>
<h1 style="font-size:20px;"><b>Feature Selections</b></h1>
<hr style="width:165px;border:2px solid rgb(255, 174, 54)">
<div class="center-justified">

<p>Please select at least one of the features below</p>

<br>
<br>
<!-- Start Input text -->
<input type="text" id="myInput" onKeyUp="myFunction()" placeholder="Search for feature..">
<!-- End Input text -->

<br>
<br>

<input type="submit" onClick="check_all()" value = "Select All">
<input type="submit" onClick="uncheck_all()" value = "Unselect All">

<br>
<br>

<form action="{{ url_for('predict') }}" method = "post">
<ul id="myUL">
<li><label for = "f1" > <input type="checkbox" name="model1" /> Artificial Neural Network (ANN) </label></li>
<li><label for = "f2"> <input type="checkbox" name="model2" /> Anserini </label></li>
<li><label for = "f3"> <input type="checkbox" name="model4" /> Bayesian Network </label></li>
<li><label for = "f4"> <input type="checkbox" name="model5" /> Best Matching (BM) </label></li>
<li><label for = "f5"> <input type="checkbox" name="model6" /> Baseline </label></li>
<li><label for = "f6"> <input type="checkbox" name="model7" /> Borda Count </label></li>
```

Figure 28. *Sample code 3 of prediction model web page in template/index.html*

9 References

Carr, S. (2020, February 13). What is remote usability testing? [Web blog post]. Retrieved from https://www.usertesting.com/blog/what-is-remote-usability-testing

Chen, J. S., & Hersh, W. R. (2021). A comparative analysis of system features used in the TREC-COVID information retrieval challenge. *117*. https://doi.org/10.1016/j.jbi.2021.103745

Donges, N. (2021). A Complete Guide to the Random Forest Algorithm. Retrieved from https://builtin.com/data-science/random-forest-algorithm

Luis Amoros, J. (2021). 5 Ways to Fight Overfitting. Retrieved from https://www.krasamo.com/five-ways-to-fight-overfitting/

Morales, J. (2020). Remote Usability Testing 101 & How to Get Started. Retrieved from https://xd.adobe.com/ideas/process/user-testing/remote-usability-testing/

Pferiffer, A. (2019). Comparing Grid and Randomized Search Methods in Python. Retrieved from https://betterprogramming.pub/comparing-grid-and-randomized-search-methods-in-python-cd9fe9c3572d

Sagar, A. (2019). 5 Techniques to Prevent Overfitting in Neural Networks. Retrieved from https://www.kdnuggets.com/2019/12/5-techniques-prevent-overfitting-neural-networks.html

Trehan, D. (2020). Why Choose Random Forest and Not Decision Trees. Retrieved from https://towardsai.net/p/machine-learning/why-choose-random-forest-and-not-decision-trees

Soni, S., & Roberts, K. (2021). An evaluation of two commercial deep learning-based information retrieval systems for covid-19 literature. *Journal of the American Medical Informatics Association*, *28*(1), 132-137. https://doi.org/10.1093/jamia/ocaa271

TREC-COVID Organizers. (n.d.). *TREC-COVID Information Retrieval*. Kaggle. https://www.kaggle.com/c/trec-covid-information-retrieval

Wang, L. L., & Lo, K. (2021). Text mining approaches for dealing with the rapidly expanding literature on COVID-19. *Briefings in Bioinformatics*, *22*(2), 781-799. https://doi.org/10.1093/bib/bbaa296

10 Annex

Team members' contribution :

Table 3. *Team members' contribution*

Team members	**Percentage**
Ong Jeng Quan	50%
Ong Meng Lap	30%
Ong Meng Chuan	20%