**Q1 & 2.**

**Paper Context –**

 There has been a global increase in hate speech, mounting several extreme violence. The law of some countries describes hate speech as speech, gesture or conduct, writing, or display that incites violence or prejudicial action against a protected group, minority, or individually based on their membership of the group.

**The current state of hate speech detection –**

Self-regulated by social media: Self-imposed definitions, guidelines, policies. Responses are generally reactive, i.e., a problem dealt with after victim complaints—human moderators.

Social media platforms like Meta, Twitter, and YouTube are overburdened by the rapid increase in the investigation of sensitive issues online and making it tougher to resolve the legal problems with government bodies. Analyzing slang and expressions across cultures, languages, and regions require a more robust state-of-the-art solution based on cutting-edge artificial intelligence techniques.

The paper "Hate speech detection using static BERT embeddings" authored by Gaurav Rajput and co-authored by Narinder Singh punn, Sanjay Kumar Sonbhadra, and Sonali Agarwal have worked their way towards contributing a method in reducing false positives in hate speech detection. Using a combination of multiple DNN models trained on the ETHOS dataset, a performance analysis is drawn by integrating word embeddings with static BERT embeddings (BiLSTM + static BE), attaining a significant increase of 8.72%.

The trained model with millions of parameters effectively detects the most challenging hate speech violations by analyzing different forms of complex content like images and videos.

**Technical Gap Identified –**

* The distinction between sub categories
* Dataset mostly English
* Small Data Size – Laborious Data Labelling
* Varied ML approach not converging to a solution
* Bias in Dataset Annotations

A fine-tuned BERT often outperforms other state-of-the-art deep neural networks in the same natural language processing test, as has been noticed when researchers first started employing BERT for these tasks, as demonstrated by the gap in past research efforts emphasized in the publication.The results of Mollas' experiment supported this conclusion.

The experiments in the study test how well-tuned BERT works when paired with other deep learning models, and they use this as their inspiration.

Providing a combination of DNNs with static BERT embedding increases the performance of hate speech detection.

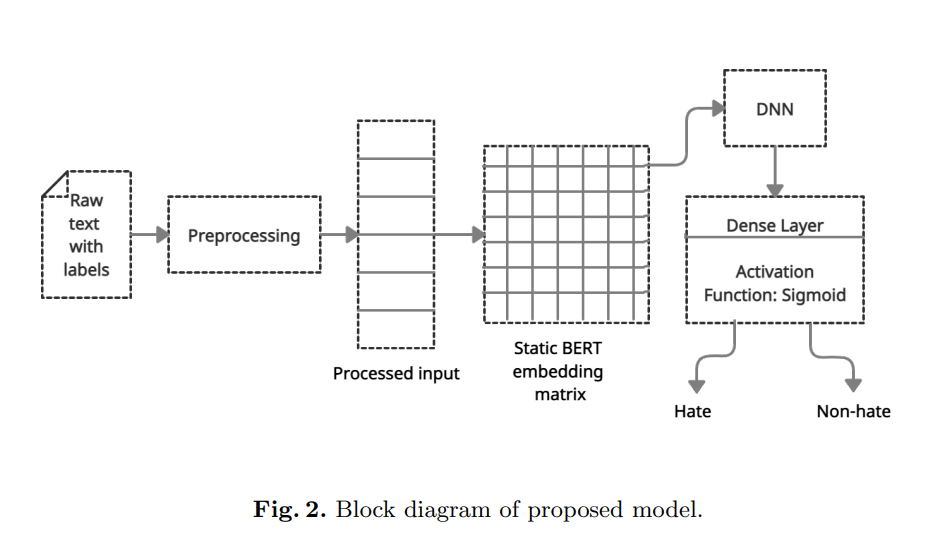
**Problem Statement Addressed in the Paper –**

 The problem is to increase the **performance of the hate speech detection classifier trained** on the ETHOS hate speech detection dataset by replacing and integrating static BERT embeddings.

**Plan of Action and Methodology –**

The proposed method emphasizes the role of BERT-based embedding in the hate speech detection framework by combining static BERT embedding with DNNs to retrieve contextual information.

To create the static BERT embedding matrix, which depicts the embedding for each word in the dataset, a sizable corpus of the dataset was first employed. The presence of hate is then determined by utilizing DNN classifiers to analyze this matrix.



The embedding matrix contains an embedding for each word in the dataset. Each row of the matrix contains the embedding for a separate word.

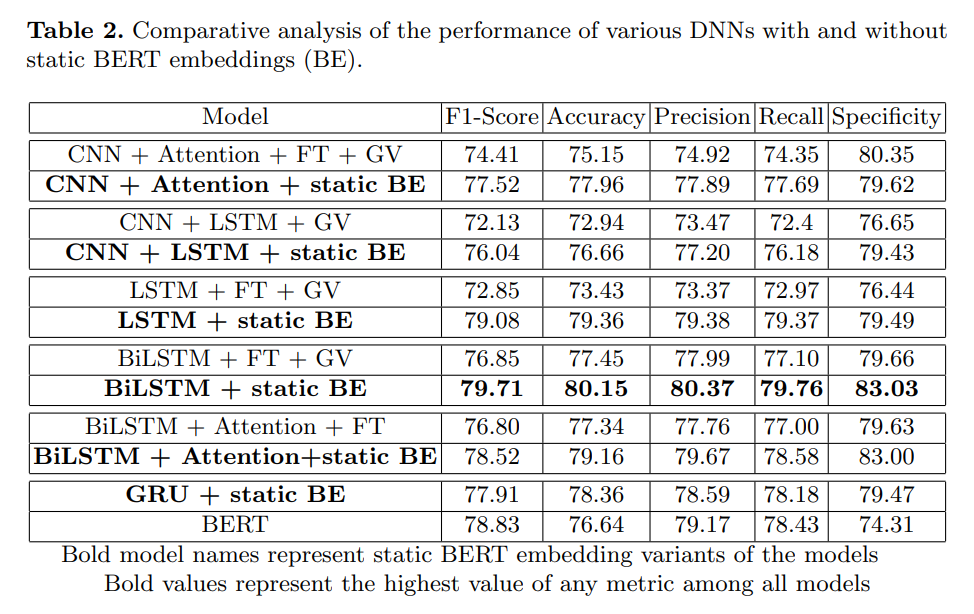
Natural Language expressions are converted into vectors and sent in fixed dimensions to the DNNs (static). BERT contextually embeds each word based on how it appears in a sentence (hence the same word has different embeddings depending on the usage context). Static word embeddings give each word its own static embedding regardless of the context in which it is used. The dictionary can hold every contextualized embedding for every word by pushing embeddings into the vector corresponding to each unique word.One can determine the static BE of a word by computing the mean of the vector holding the word's contextualized BERT embedding. For terms not in the lexicon, BERT creates their embeddings and breaks them down into subwords.

The embeddings of subwords are employed to build the embedding of a phrase that wasn't in the vocabulary.

Finally, they create the embedding matrix using Keras Tokenizer and static BERT embeddings.

The authors have referenced previous related work on the topic and tried a fine-tune BERT model with a combination of BiLSTM and static BERT embeddings, which outperform all existing models.

**Results and Findings –**

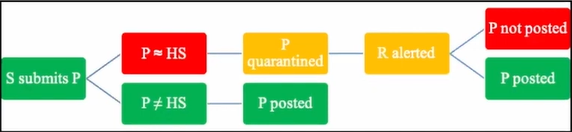


The results on various DNNs explain that deep neural networks with static BERT embeddings outperform the same deep neural networks that use embeddings as fastText, GloVe, or fastText + GloVe in all metrics. DNNs like CNN using attention LSTM CNN + LSTM, BiLSTM and BiLSTM using attention the average (avg) increase in F1-score is 3.56%, accuracy is 3.39%, precision is 3.40%, recall is 3.55% and sensitivity is 2.37%. Hence, it is evident that static BERT embeddings provide better feature representation than fastText, GloVe, or fastText + GloVe.

Furthermore, BiLSTM using static BERT embeddings (BiLSTM + static BE) performs better in all metrics than other DNNs under consideration.

**Expected Answer –**

Hate speech is an advanced research area, and new development is in process. One such possible answer is implementing methods of quarantining hate speech and isolating hate words in a way identifying offensive posts reaching a larger audience.



**References –**

1. Rajput, G., Singh punn, N., Sonbhadra, S. K., and Agarwal, S., “Hate speech detection using static BERT embeddings”, arXiv e-prints, 2021.
2. Deep learning for hate speech identification in tweets: Badjatiya, P., Gupta, S., Gupta, M., and Varma. Pages 759–760 of the book Proceedings of the 26th International Conference on World Wide Web Companion (2017)
3. Neural machine translation by simultaneously learning to align and translate, D. Bahdanau, K. Cho, and Y. Bengio. Preprint: 1409.0473, arXiv (2014)
4. Automated hate speech identification and the issue of objectionable language. Davidson, T., Warmsley, D., Macy, and I. Weber. International AAAI Conference on Web and Social Media Proceedings, vol (2017)
5. Ethos: an online hate speech identification dataset, Mollas, I., Chrysopoulou, Z., Karlos, S., Tsoumakas, G. Preprint accessed at arXiv:2006.08328 (2020)
6. Allan, R. (2017, June 27). Hard questions: Who should decide what is hate speech in an online global community? Facebook Newsroom. Retrieved January 28, 2019 from https://newsroom.fb.com/news/2017/06/hard-questions-hate-speech/.
7. Ullmann, S., Tomalin, M. Quarantining online hate speech: technical and ethical perspectives. Ethics Inf Technol 22, 69–80 (2020). <https://doi.org/10.1007/s10676-019-09516-z>

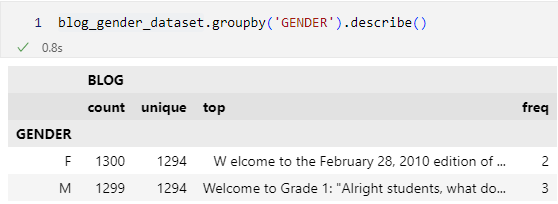
**Q3.**

**Business Understanding**

The problem statement is to identify and detect the gender of bloggers based on a corpus of textual data to evaluate and determine gender for potential usage of Market Research and Gender Targeting for personalized content recommendation.

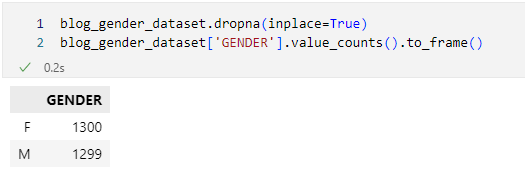
**Data Understanding**

The dataset consists of two columns: "BLOG" and "GENDER."

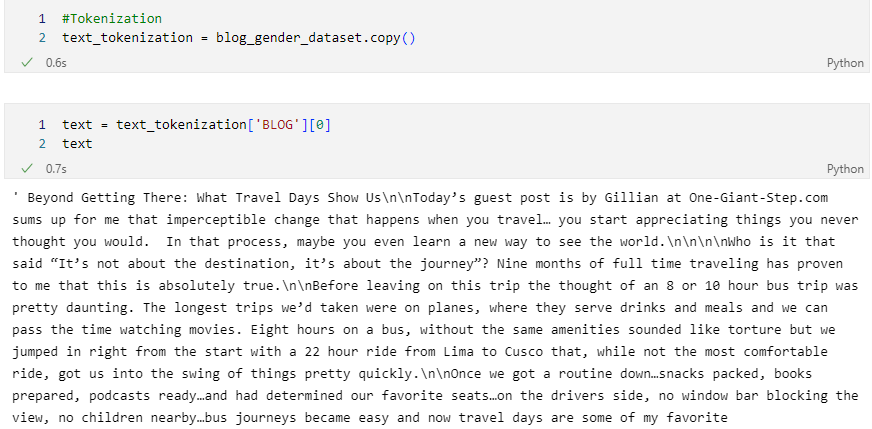
There are a total of ~2599 records with 1300 Females and 1299 Males.

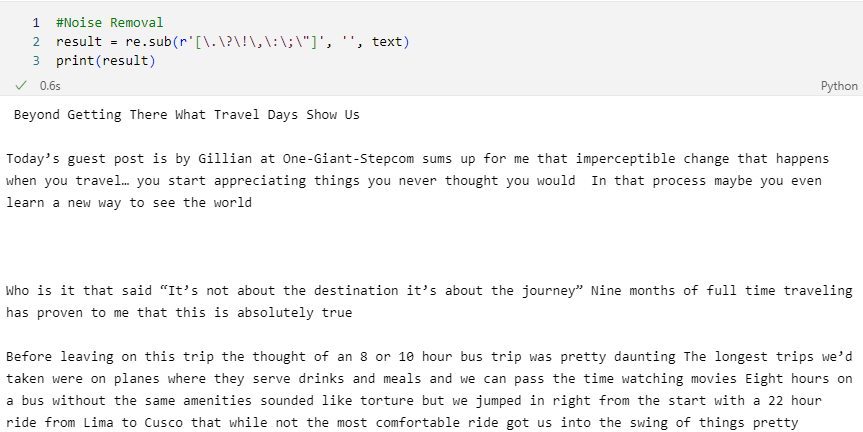
**Data Preparation**

The raw data was analysed using statistical analysis. Nulls were identified and dropped as it was insignificant to the model.

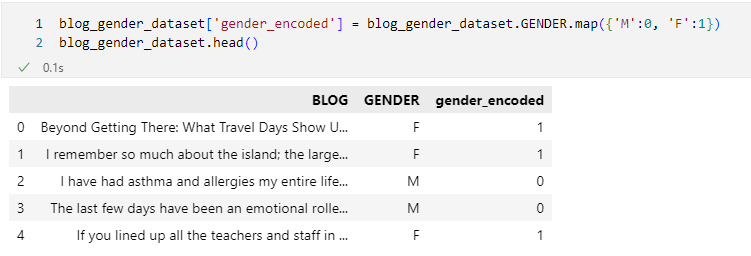


Exploratory data analysis comprised of Tokenization and Regex was also used to remove symbols and noise from the textual dataset.

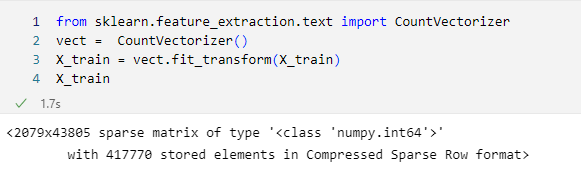


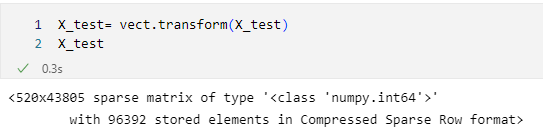


The gender column was mapped to 0 & 1 respectively for Males and Females.



CountVectoriser was used to transform the BLOG training data into vector matrix. It implies a sparse representation of the data.





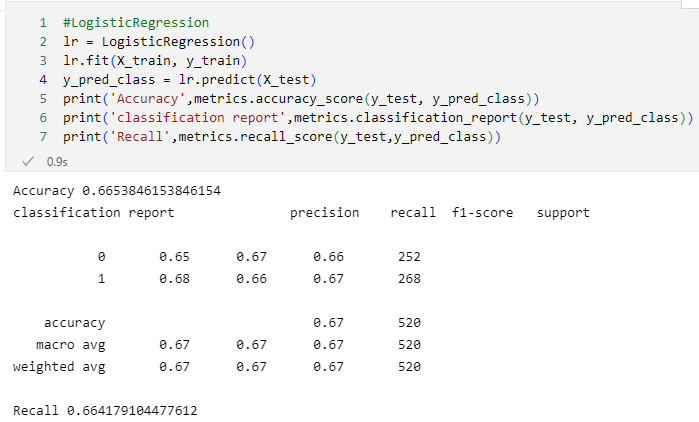
**Modelling**

After data was processed it was split into 80:20 ratio and the dataset was further trained onto 7 different machine learning algorithm.

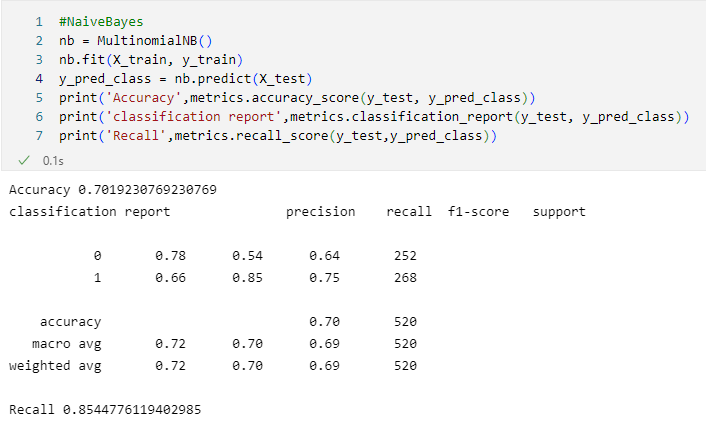
1. Logistic Regression
2. Naïve Bayes
3. Support Vector Machine
4. K-Nearest Neighbours
5. Decision Tree
6. Random Forest
7. Multi-Layer Perceptron Classifier

**Evaluation**

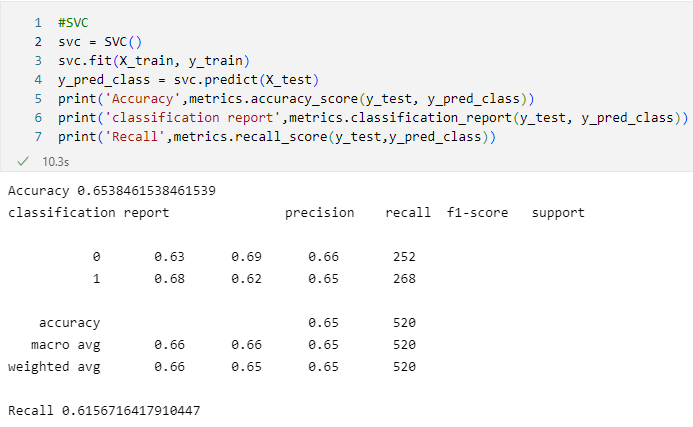
1. Logistic Regression



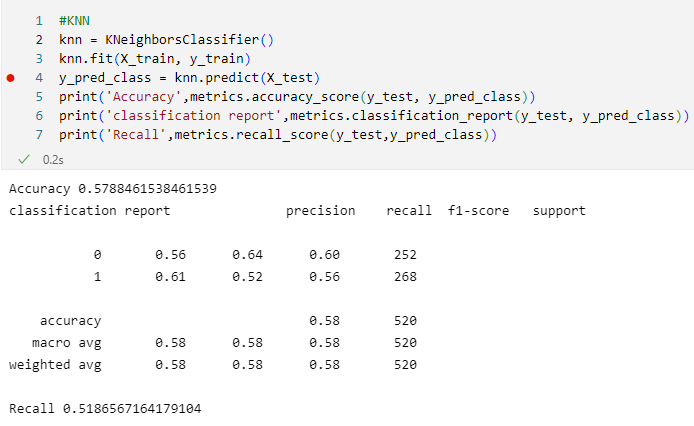
1. Naïve Bayes



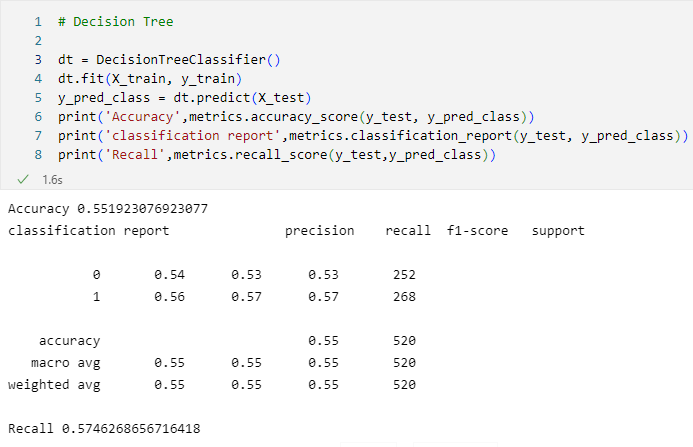
1. Support Vector Machine



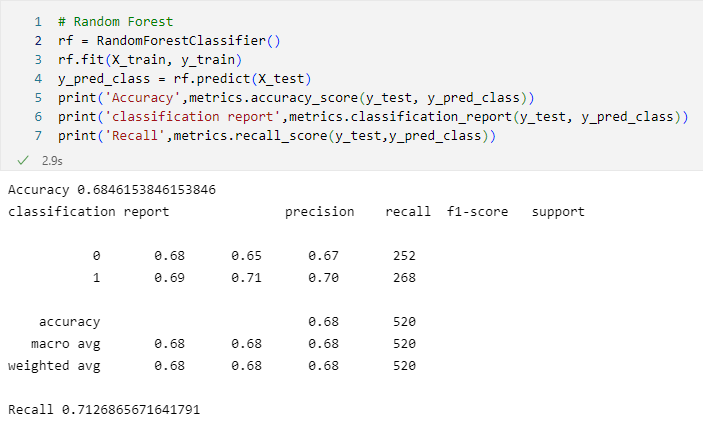
1. K-Nearest Neighbours



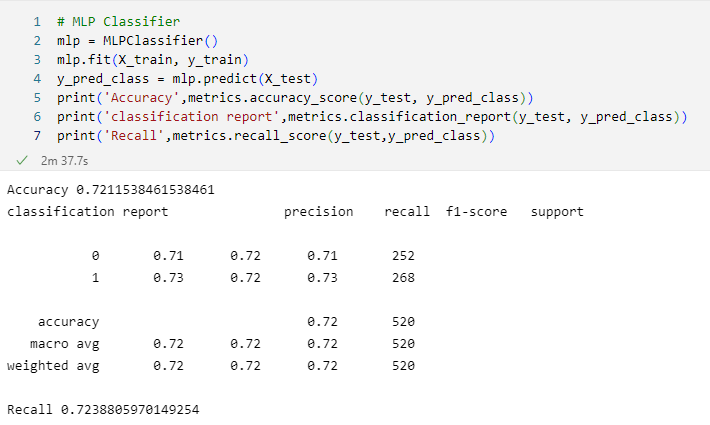
1. Decision Tree



1. Random Forest



1. Multi-Layer Perceptron Classifier



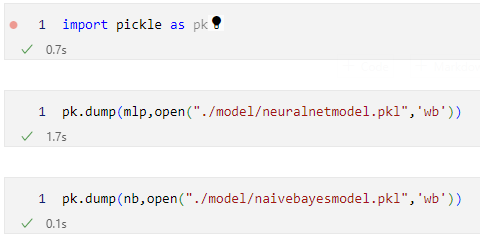
In this performance comparison Naïve Bayes and Multi-Layer perceptron are two of the best performing models. Where Naïve Bayes has an accuracy of 70% but relatively a higher recall rate at 85%, whereas in MLP the accuracy is 72% which is 2% better than the Naïve Bayes and recall stood at 72%.

In my verdict, I would choose **Naïve Bayes if the data size is less**. And in case the **dataset is more I would look forward to deploy MLP model into a production** given it will outperform the existing metrics with more data in place.

**Deployment**

I have used pickle to save the performance of my model as a checkpoint so I can load the model from there to make inference.

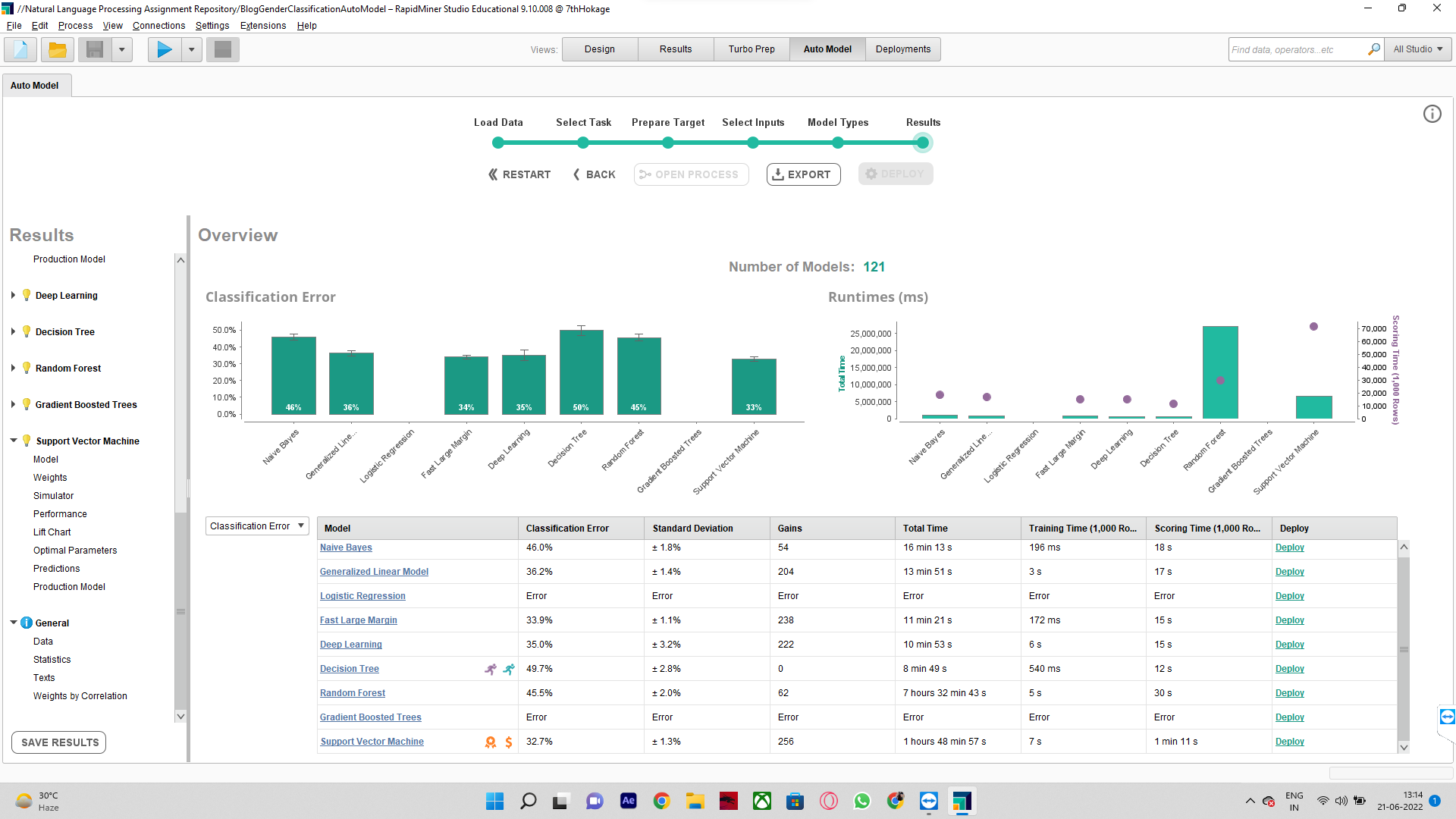
Now the pickle file can be deployed in multiple ways as loading it on Flask/Heroku/StreamLit app or consuming as a REST API endpoint.



**RapidMiner Results**

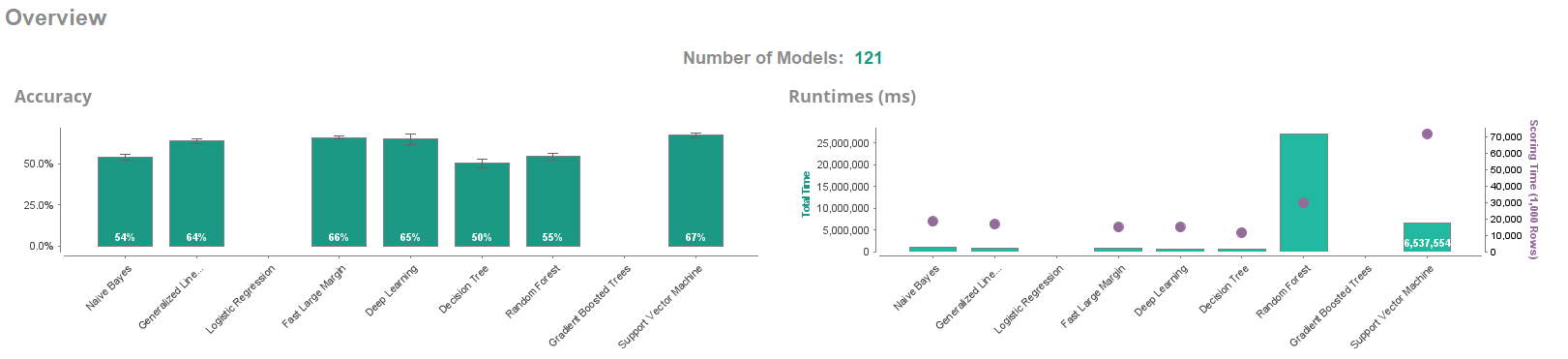
**Classification Overview**

The below auto-model is an extension on RapidMiner which helps to build predictive model end to end from loading raw data pre-processing to modelling and deployment. On running a benchmark test on the auto model following results have been achieved based on gender classification on blog posts.



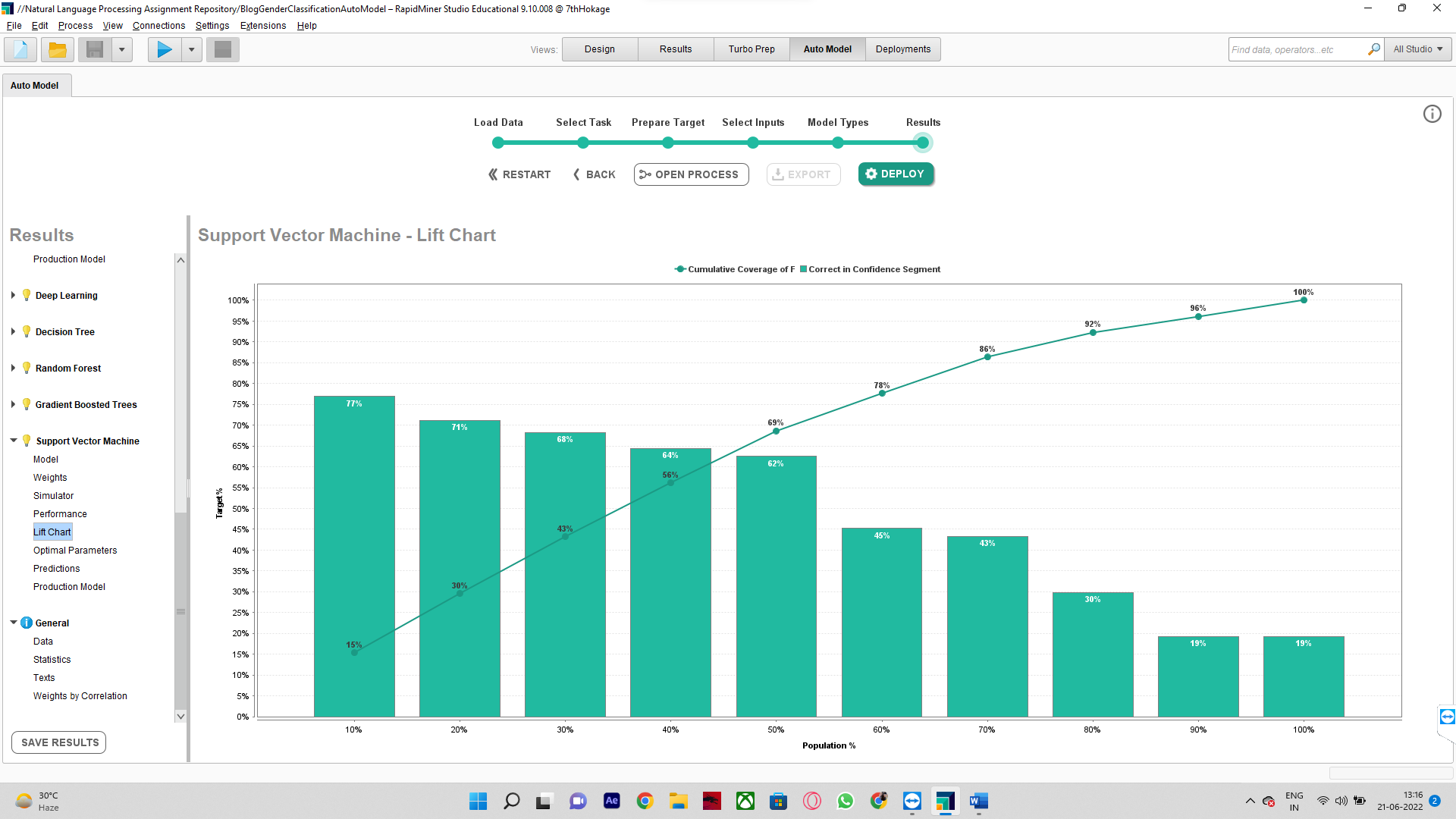
It's evident from the results above that the best model suggested by the application is **Support Vector Machine** with the least **Classification Error – 32.7%** at the same time **Decision Tree is a faster model** with runtime of 8mins 49 sec but **takes a hit in the accuracy** of the prediction as a trade-off having highest classification error.

Note: Due to technical glitches and heap issues Logistic Regression and Gradient Boosted Trees were not able to finish modelling and had to skip in the comparison.



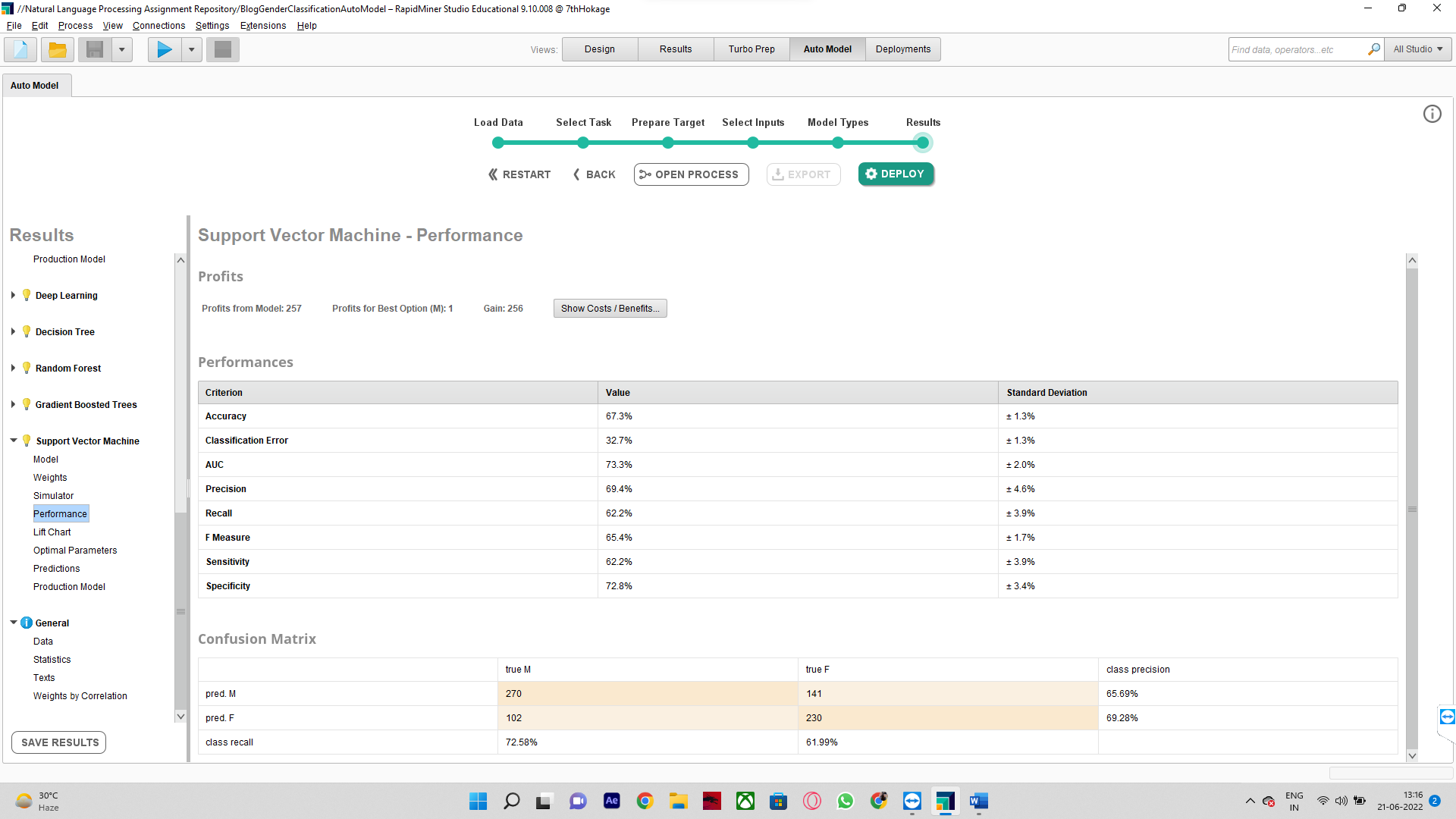
The below comparison depicts the 10 bins on the test data with a decreasing confidence score. Where the highest confidence values are in the first bin and so on. If we see the centre of the chart, we can see that the model would correctly classify 62% of the target data with only using 50% population data.

Using this representation, we are able to understand how our Support Vector Classifier model is compared against a random guess and translating the difference in lift score.

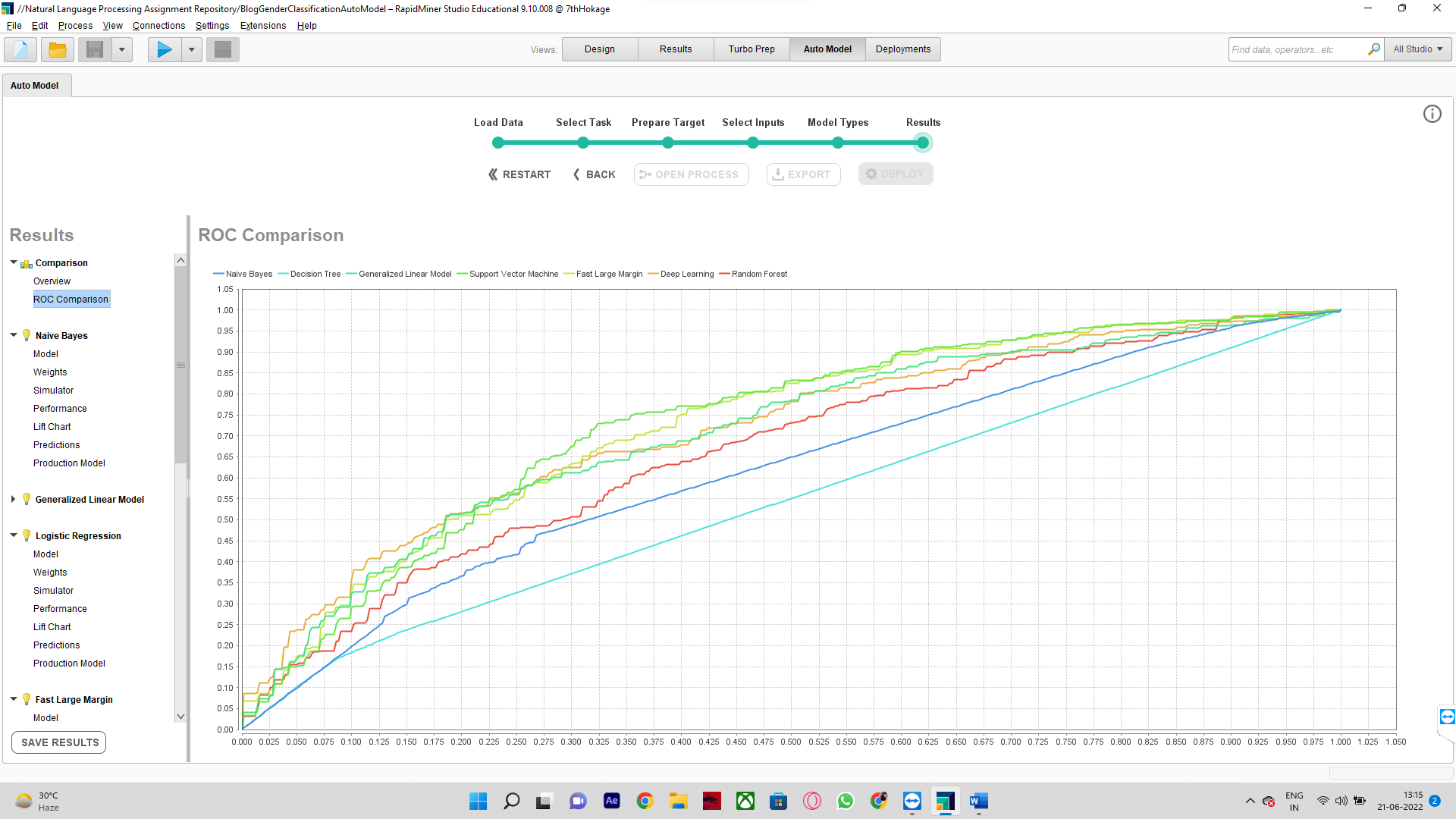


From the below screenshot we can observe a list of performance criteria for our best Classifier – SVM.

The results don’t seem to be tuned but can perform better in real life scenarios with proper data cleansing and data mining.



The below ROC curve depicts the classification efficiency of all the model trained using RapidMiner’s Auto-modelling. On the x-axis there are false positives (FPR) and true positives in the y-axis for all 7 models respectively.



**Performance Comparison**

Overall fine-tuned Machine Learning models still outperforms RapidMiner results. On the contrary, models trained on RapidMiner eases up the model development time which is a noticeable profit so that a Data Scientist or Artificial Intelligence Engineer can better spend time in preparing domain understanding and ways to improving optimization and performance rather than spending time into building redundant time-consuming models.