AI Bible App

Gabriel Villasmil (sba23199@student.cct.ie)

CCT College Dublin

Abstract

The more than 31,000 verses makes reading the bible an overwhelming task for individuals that does not know where to start reading the Scriptures or how can actually those verses be relevant to their current spiritual journey. Specially the youth, they are less involved in church.

The AI-Powered Bible Study Personalization App addresses these issues by providing users with Machine Learning trained models in recommending Bible verses and the new demanded usage of LLM; AI-generated commentary specifically prompted to the verse. We will make an impactful app by solving the problem of a personalized user engagement.

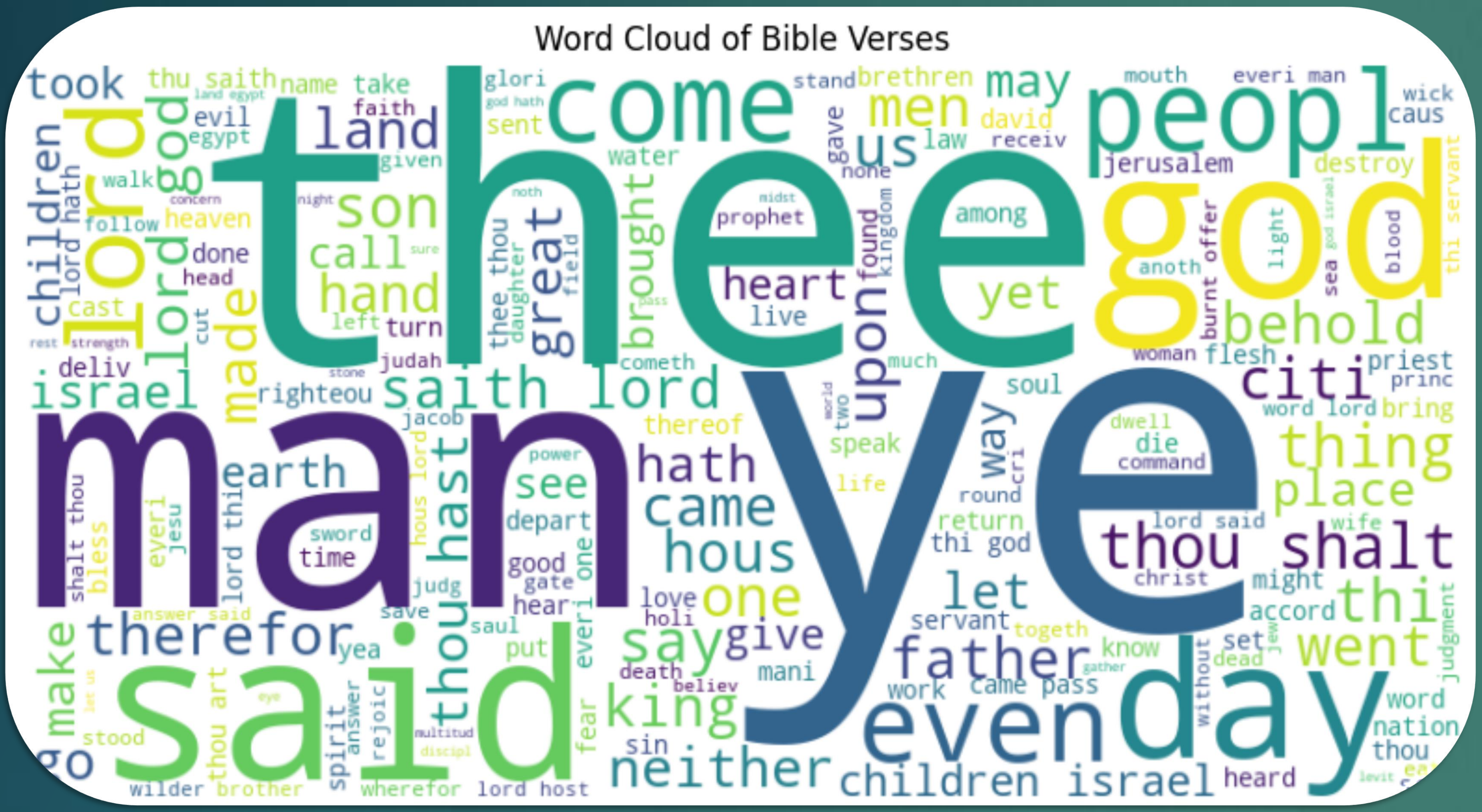
The church that takes advantages of AI will have will have a impact in a younger generation, this new engagement will lead this church to grow, which means potentially increasing their donations.

Objectives

By leveraging machine learning algorithms, text classification models, for categorizing Bible verses into given topics (E.G: Love, Salvation, Trust, Guidance, Relationships, and Health) from the King James Bible (KJV) the goal is to train a machine learning model that can predict accurately the bible verse based on the given Topic.

Research

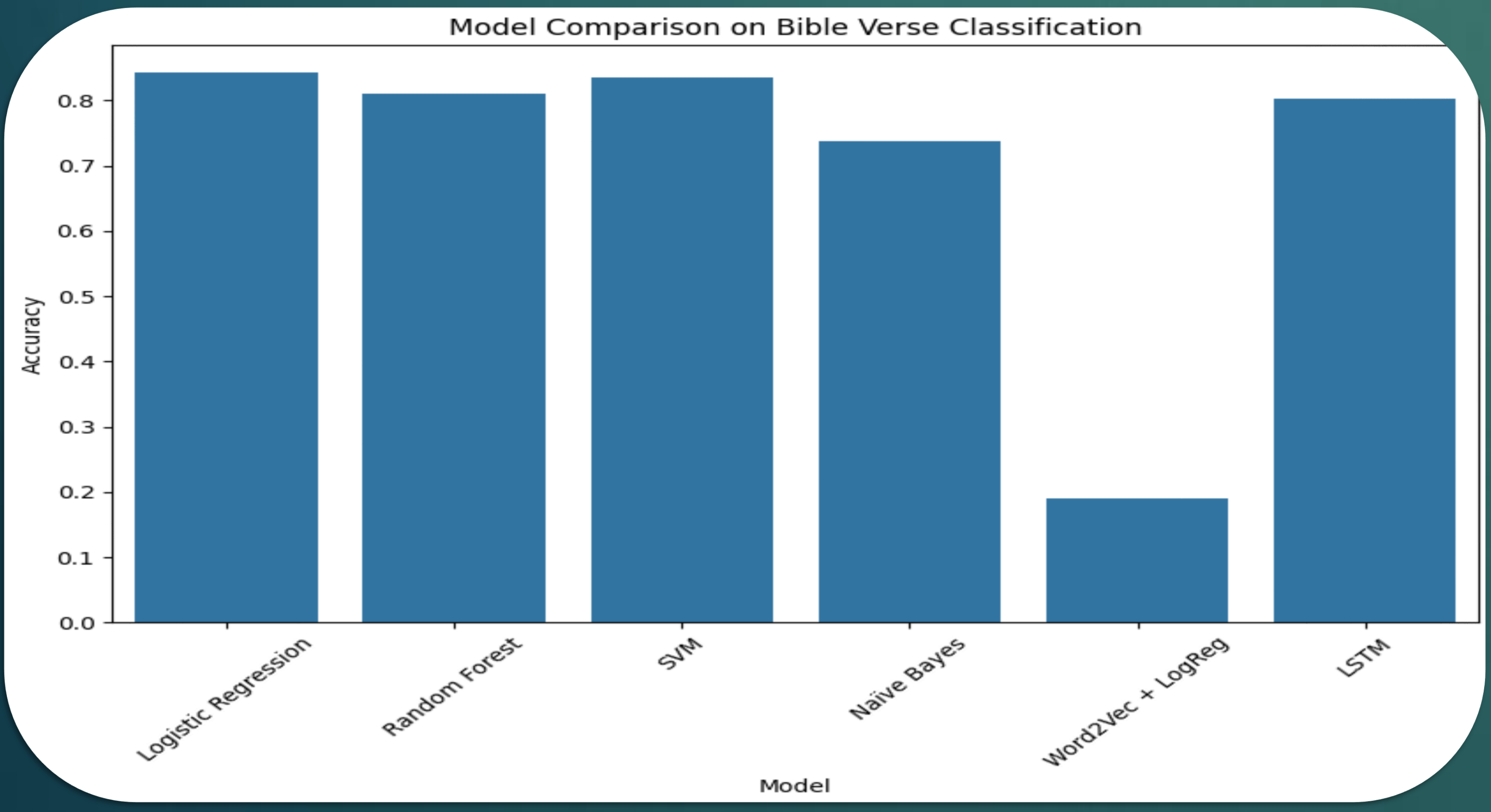
Using the whole bible as dataset was thanks to Kaggle, identifying the most common words and getting grid of them, was a crucial part for continuing with this study. Using the NLTK library, a function was created to tokenize words, remove stop words and stemming words.



When the data was ready for machine learning, the next step was to train the dataset. A new training dataset was webscrapped for this task thanks to BeautifulSoup Python library from the DailyVerses.net webpage. Passed our just created function and used it for Machine Learning testing.

After that, a wide range of different models were specifically selected to complete this task:

Important to mention, vectorizing was used for all the models first whether it was Word2Vec or TF-IDF. TF-IDF stands for Term Frequency Inverse Document Frequency of records. I define as the calculation of how relevant a word in a series, in a dataset in this case is to a text. The importance increases to the number of times in the text the word appears and is compensated by the word frequency in the corpus, the dataset.

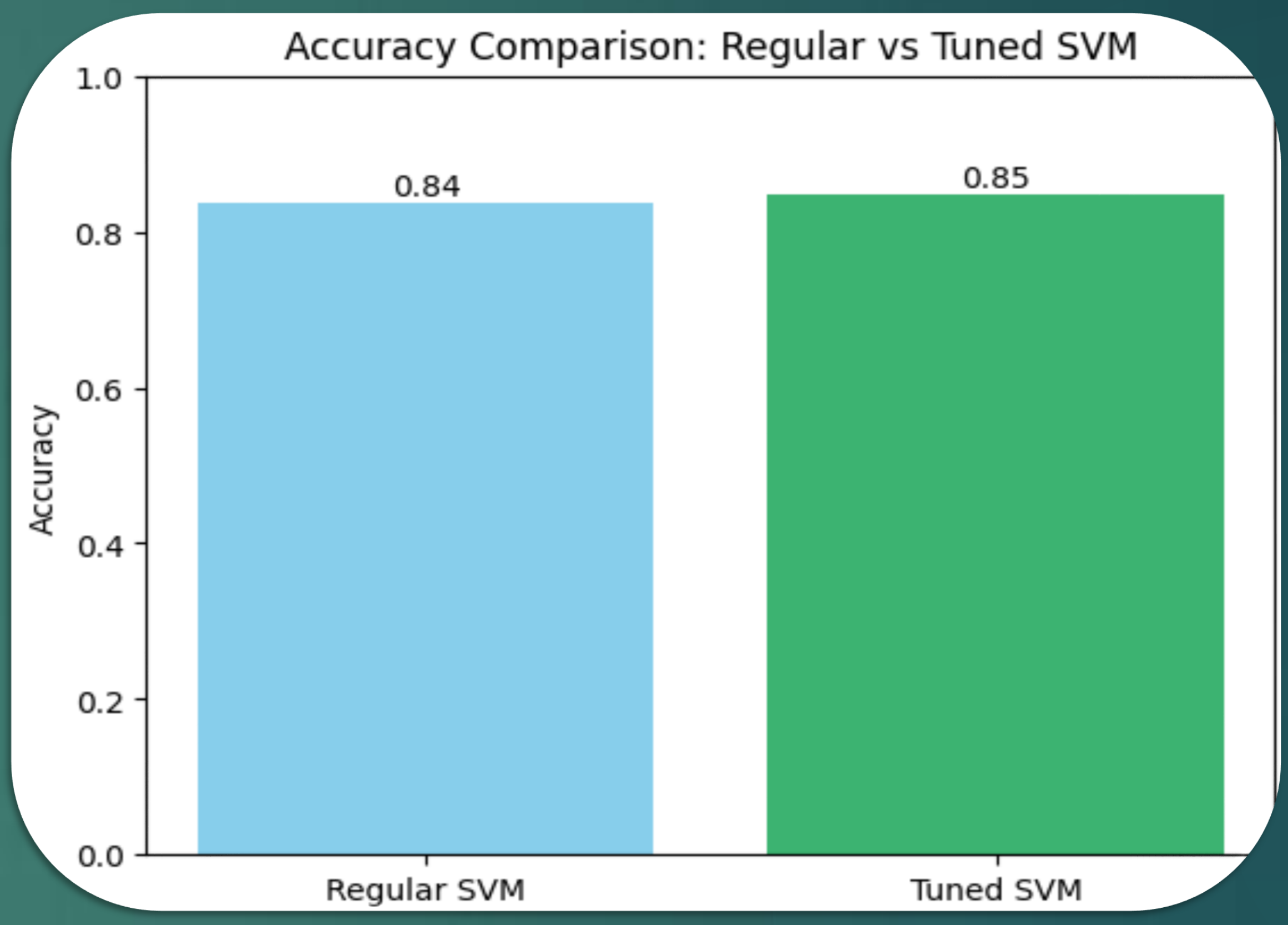


Conclusion

Let's make clear that this was an almost one-year academic project, during which time I trained multiple algorithms, multiple times, even much more than the ones that made it to this report, with that being said there were some algorithms that for some reason after time they performed differently of what they previously performed. One example was the Random Forest, from time to time, it performed better than my best performer, and when I was about to finish this project, out of the blue, the Logistic Regression started to perform better than all. Based on this understanding, I chose the model that not only gave me the best results in the accuracy test but also the one that was consistent in giving me those results over time, throughout this project.

SVM and Tuned SVM

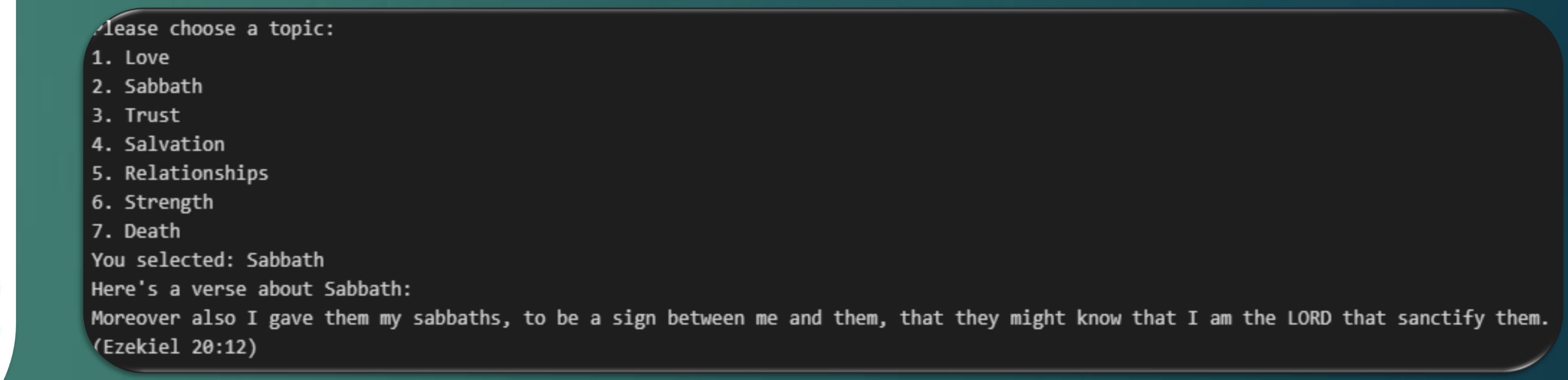
"SVMs are commonly used in natural language processing (NLP) for tasks such as sentiment analysis, spam detection, and topic modeling. They lend themselves to these data as they perform well with high-dimensional data." (IBM, 2023)



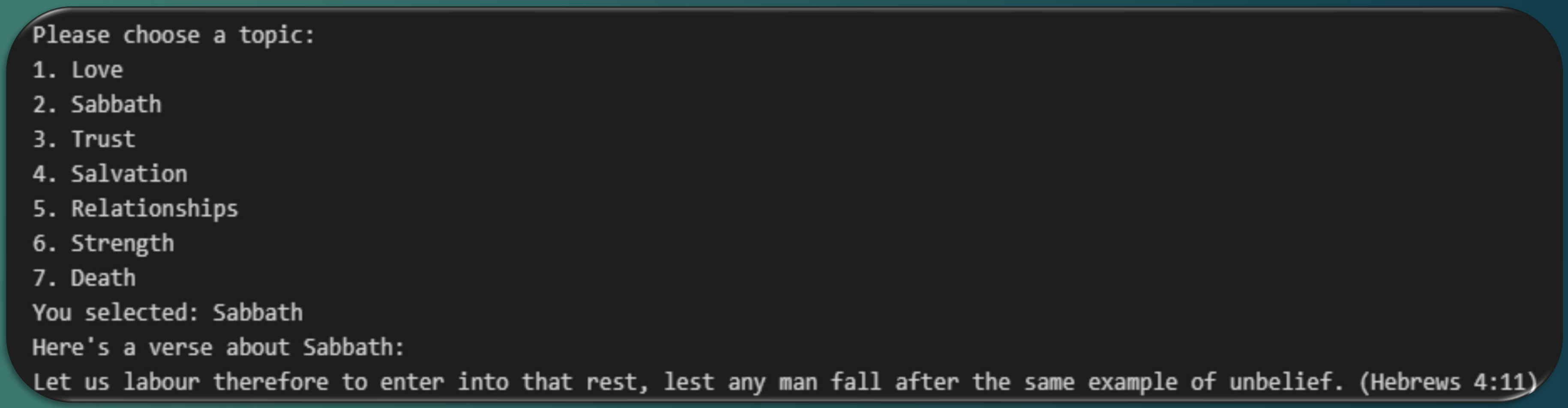
I would say this is my favourite mid-level model. Now, the fact that it works great in text processing is something that I learnt during this case, making it a must-use. Some parameter tuning and cross-validation techniques were added to strengthen even more this powerful model and compare it with the plain version.

Accuracy test: Regular: 0.85 and sometimes 0.84. Tuned: 0.85, by a small difference, this is the best model performer found until this moment. After tuning it with cross-validation techniques, it was found that it even performed better. This one steadily performs the same after multiple retraining attempts

Once the training was done, now comes the real test came, an MVP of the real app was built in the console, and Tuned SVM model was fit into the original dataset. These are the results: Let's see two predictions about the Sabbath topic:



```
Please choose a topic:
1. Love
2. Sabbath
3. Trust
4. Salvation
5. Relationships
6. Strength
7. Death
You selected: Sabbath
Here's a verse about Sabbath:
Moreover also I gave them my sabbaths, to be a sign between me and them, that they might know that I am the LORD that sanctify them.
(Ezekiel 20:12)
```



```
Please choose a topic:
1. Love
2. Sabbath
3. Trust
4. Salvation
5. Relationships
6. Strength
7. Death
You selected: Sabbath
Here's a verse about Sabbath:
Let us labour therefore to enter into that rest, lest any man fall after the same example of unbelief. (Hebrews 4:11)
```

As we can see in this second prediction, this algorithm works great in finding related context verses about the Sabbath without the need for the root word in the text itself.

Without a doubt, I would proceed with the creation of this app; a church that leverages the use of IT will get a huge competitive advantage. First of all a church that uses AI will be more attractive to a younger generation that was born with an understanding and the importance of it. This app will boost people's understanding of the Word of God, attracting more people to look to spend time and congregate in this cool local church, helping this church to grow in numbers and finally, potentially increasing the amount of donations

References

CodeSignal (2024). Mastering Random Forest for Text Classification. [online] CodeSignal Learn. Available at: https://codesignal.com/learn/courses/introduction-to-modeling-techniques-for-text-classification/lessons/mastering-random-forest-for-text-classification.

DailyVerses.net. (2025). Bible Verses by Topic. [online] Available at: https://dailyverses.net/topics [Accessed 5 May 2025].

IBM (2023). Support Vector Machine. [online] IBM. Available at: https://www.ibm.com/think/topics/support-vector-machine.

MathWorks (2025). Classify Text Data Using Deep Learning - MATLAB & Simulink. [online] Mathworks.com. Available at: https://www.mathworks.com/help/textanalytics/ug/classify-text-data-using-deep-learning.html [Accessed 5 May 2025].

Otten, N.V. (2023). Tutorial TF-IDF vs Word2Vec For Text Classification [How To In Python With And Without CNN]. [online] Spot Intelligence. Available at: https://spotintelligence.com/2023/02/15/word2vec-for-text-classification/.

Turing (2025). How to Use Naive Bayes for Text Classification in Python? [online] www.turing.com. Available at: https://www.turing.com/kb/document-classification-using-naive-bayes.

Geeksforgeeks (2024). Removing stop words with NLTK in Python. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/removing-stop-words-nltk-python/.