

Contents

Introduction 4

Problem Definition 4

[Business Understanding 4](#_TOC_250008)

[Strategic Overview of the Business Problem 4](#_TOC_250011)

[Objectives 5](#_TOC_250007)

[Scope 5](#_TOC_250009)

[Project Management Methodology: CRISP-DM Framework 6](#_TOC_250007)

[Ethical Considerations 6](#_TOC_250009)

[Project Plan 6](#_TOC_250009)

[Technologies Used 7](#_TOC_250007)

[Dataset Overview 8](#_TOC_250006)

[Dataset Understanding 8](#_TOC_250005)

[Dataset Preparation 11](#_TOC_250004)

[Removing Stop Words 11](#_TOC_250003)

[Tokenization 11](#_TOC_250002)

[Stemming 11](#_TOC_250001)

[Web Scrapping Training Data 12](#_TOC_250004)

[Machine Learning implementation 13](#_TOC_250009)

1. [Model Training and Evaluation 1 13](#_TOC_250008)

2. [Model Evaluation 2 and Conclusion: Accuracy Scores 17](#_TOC_250007)

3. [Deployment: Bible Verse Recommendation System 18](#_TOC_250007)

[Challenges 19](#_TOC_250006)

[Conclusion 20](#_TOC_250005)

[References 21](#_TOC_250000)

[GitHub Link 23](#_TOC_250000)

[Presentation Link 23](#_TOC_250000)

[Dataset Link 23](#_TOC_250000)

Introduction

Final attempt with the dataset of my Bible app, in this third CA I will actually not only take a deep into the dataset, but now I am going to present some amazing results and findings.

Problem Definition

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.

This lack of personalized guidance makes it difficult for individuals to maintain consistent engagement with Scripture making them drastically stops their spiritual growth. Furthermore, while many apps and online resources exist with devotionals for growing in understanding of the Scriptures, few offer, if any, this kind of customized AI-driven recommendations.

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.

Business Understanding

What we need to do is **classify Bible verses** into specific themes and recommend them to our users, the rest is noise. Keys include:

* Categorization: Bible verses into six themes—Love, Salvation, Trust, Guidance, Relationships, and Health.
* Identifying stakeholders: individuals seeking their spiritual guidance and a Bible study.
* Establishing success criteria: A high-performing classification model with excellent metrics in precision, recall, and F1-score.

Strategic Overview of the Business Problem

The Natural Language Processing (NLP) field has experienced rapid advancements nowadays; applications going from sentiment analysis to machine translation. So I took advantage of that in this fascinating domain where NLP can be used in the regards of religious texts, specifically the Bible. The Bible being a rich source of spiritual wisdom, and its verses cover a wide array of topics such as love, trust, salvation, health, guidance, etc and given the immense relevance of the Bible in religious and spiritual practices around the world due to its growth, being able to automatically categorize and recommend Bible verses based on themes is incredibly valuable for both personal growth and educational purposes.

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 correct topic of a Bible verse based on its content. 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

This project addresses the challenge many Christians face when seeking meaningful notes for their own cases. Through this innovative approach, the app aims to make Bible study more aligned with individual growth. Main goals:

Incorporate a Variety of Spiritual Themes: Making sure that our app covers a broad range of topics for all users’ spiritual journeys.

Accuracy in recommendations: Give people the new bible verse they are looking for.

Different ways of recommendations: By having the user to choose from his own history of verses, top bible verses of the app or verses by a selected topic.

Engagement: Creating an engaging platform where users can spend time having a relationship with God.

Creating a deeper understanding of the Scriptures: AI-generated commentary will help in the understanding of the selected verses and the new ones.

Ethical AI Use: we will make sure that the ethical use of AI in generating personalized recommendations and spiritual insights following norms.

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 correct topic of a Bible verse based on its content.

Text preprocessing, feature extraction, model training, and evaluation are implementations given in this project to achieve this. Firstly, our first choice is the model **Logistic Regression**, one of the simplest as well as effective classification algorithms. Logistic results were given already in the second attempt but I found out in this third attempt a better algorithm to predict these bible verses.

When the app is finished, we can promote it to local churches which will make people have more interactions and engagement with the church, leading them to have a growing church organization, better relationships and boosting donations.

Scope

The Natural Language Processing (NLP) field has experienced rapid advancements nowadays; applications going from sentiment analysis to machine translation. So I took advantage of that in this fascinating domain where NLP can be used in the regards of religious texts, specifically the Bible. The Bible being a rich source of spiritual wisdom, and its verses cover a wide array of topics such as love, trust, salvation, health, guidance, etc and given the immense relevance of the Bible in religious and spiritual practices around the world due to its growth, being able to automatically categorize and recommend Bible verses based on themes is incredibly valuable for both personal growth and educational purposes.

With that being said our main goal in this two semester project is to train the best ML algorithm for this future AI Bible App that will help people find the perfect verse for their current situation.

Project Management Methodology: CRISP-DM Framework

CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology was used in this project due to its great features, one of them involving the fact of being able to make adjustments very quickly.

Published in 1999 to standardize data mining processes across industries, it has since become the most common methodology for data mining, analytics, and data science projects.

Data science teams that combine a loose implementation of CRISP-DM with overarching team-based agile project management approaches will likely see the best results

The six phases of CRISP-DM are being applied and most likely will continue to be to this Bible verse classification and recommendation project.

Ethical Considerations

Important: The answer given by the topics of choice will be given under the Christian lens of viewing the world. GPDR Compliance registers religious data as sensitive data, every information from the users will be treated diligently. BibleData from Kaggle comes with a NonCommercial, Attribution and ShareAlike license so it is free to use, share and adapt, all that I need for this kind of project. I will provide the link to the source in references section.

Project Plan

Data Collection (2 weeks): What is a Machine Learning without data? I have in mind at least three different tables: a. Bible Verse Table: A table with all the bible verses. This one will be given by Kaggle. I personally will add different columns, one will contain a column named “Main\_Topic” and another called “Related\_Topics”. By having these two columns, first I can properly label the main topic of the verse and train the model. Second, with the later one I can let the algorithm create a correlation between verses, making those columns the heart of our app. What I just described is the main result I want to get and my first thought to reach it, but the way of achieving it, may vary along semesters. b. User Profile Table: Another crucial table, this one will track the history of each individual interactions with the app giving me valuable insights for running ML algorithms based on the “Related\_Topics” column from the Bible Verse Table. This table will be given by the very interaction with the app. c. Recommendations Table: Of course after all the machine learning process I need to provide some results. This table will give us at least an User\_ID and Recommended\_verses columns.

Data Cleaning (3 weeks): In this stage I will do data cleansing to make these new tables fit my needs for this app. As mentioned before, two columns will be created to categorize bible verses and train the model. Later on, and EDA could be done to explore patterns with my own eyes before running the algorithms, but that it is something that might not be relevant for this project, I see myself doing it but excluding it out the capstone project file.

Coding (7 weeks): This is where I will look upon internet, or the material given in class, to best Machine Learning algorithms to work with when using text. Previously, on my personal portfolio projects I worked with Cosine Similarity algorithm in a similar recommendation project. The results were accurate so I will look upon that one in first place.

Design (5 weeks): Working in a variety of projects using Machine Learning with RDMS I learned this framework for creating a fast web app for any Python script which is actually called FastAPI. In this stage I will be working more on the HTML and CSS formatting while with Python building an interface that can be used everywhere locally.

Presenting (1 week): while it is true that throughout all the process it is required to upload files on GitHub, this final stage is where everything will be arranged an prepared to uploaded to meet the deadline.

**Let’s recap briefly the roadmap for this project:**

1. Data Collection (2 weeks): Done.

2. Data Cleaning (3 weeks): Partially Done.

3. Coding (7 weeks): Partially Done. This is where I will look upon the internet, or the material given in class, to best Machine Learning algorithms to work with when using text. This is where we are right now, it is one of the longest periods due to the fact that we need to find the perfect machine learning algorithm

4. Design (5 weeks): In Progress.

5. Presenting (1 week): In Progress.

Technologies Used

**Models and machine learning algorithms**

For this project, I have used Exploratory Data Analysis, Data Preparation, and Data Visualization. The machine learning models used are Logistic Regression, Random Forest, SVM, Naïve Bayes, WordVec + Logistic Regression, and the LSTM model. All machine learning models were selected due to their nature in text processing data.

**Libraries**

Numerous libraries were used to clean, visualise, and create the machine learning models. These include: Pandas, Numpy, Seaborn, Matplotlib, Scipy, nltk, Sklearn, wordcloud, etc, including the famous BeautifulSoup for webscrapping process explained later on.

Let’s see some examples:

**Pandas** is a must when working with datasets in Python

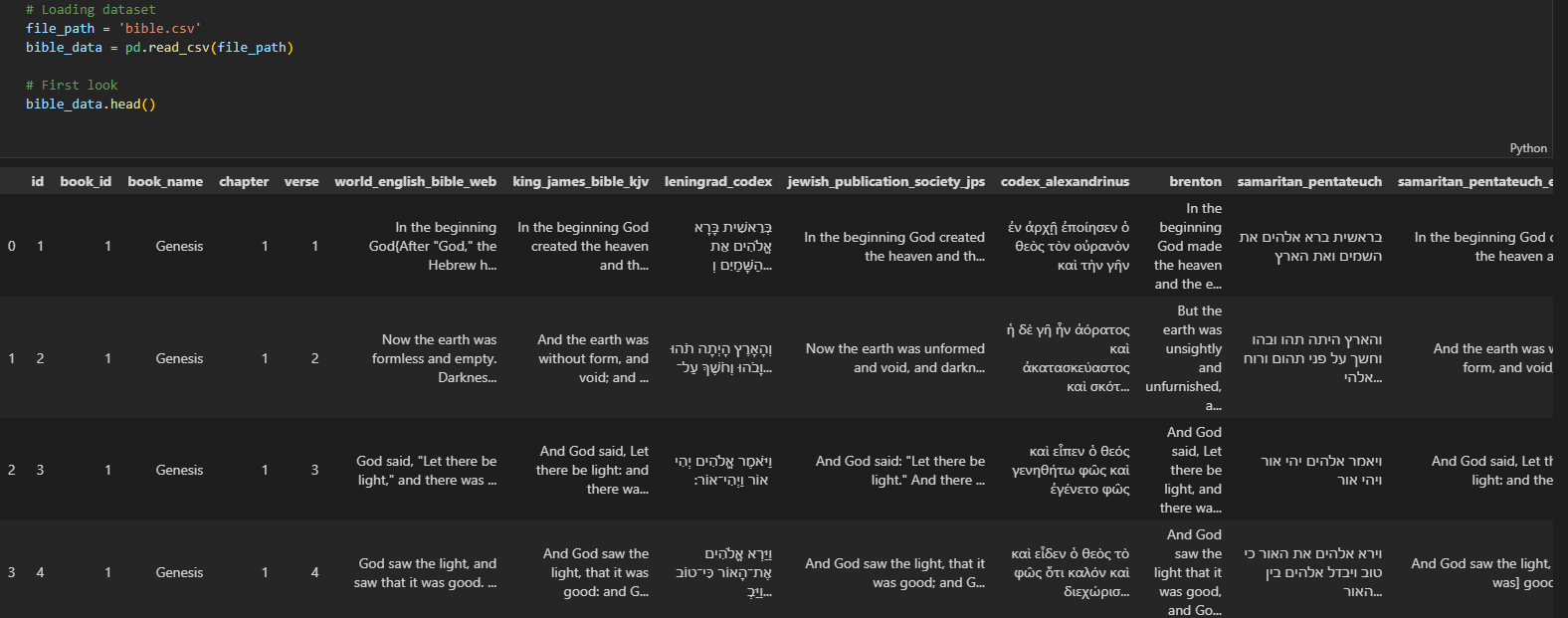
**Numpy,** if you are going to get involved in numerical data, like for example in this case, when vectorizing words to convert them into numbers to deal with them in the model development phase.

Dataset Overview

Data Sources: BibleData from Brady Stephenson will be my choice for this app. This dataset brings all the bibles verses contained in an actual English Bible in form of a table, more than suitable for running analysis or any kind of algorithms on top of it. In order to use this dataset you need to create an account in Kaggle. I will provide the link to the source in references section.

Dataset utilized in this project, thanks to a Kaggle dataset, contains the Bible verses from the original King James Bible. Secondly, from that dataset I created a small training dataset, each labeled with a specific topic. As mentioned before these are: Love, Salvation, Trust, Guidance, Relationships, and Health.

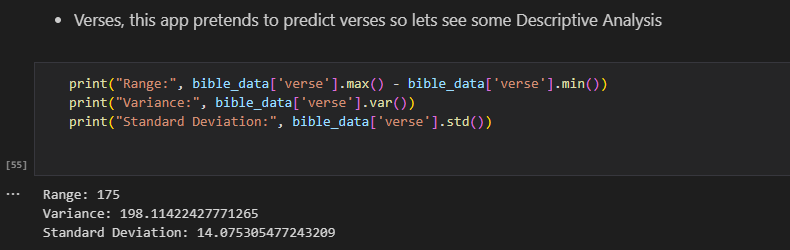
Each verse is preprocessed and transformed into a numerical format suitable for machine learning.

*Figure 1: bible\_data.head.()*

Dataset Understanding

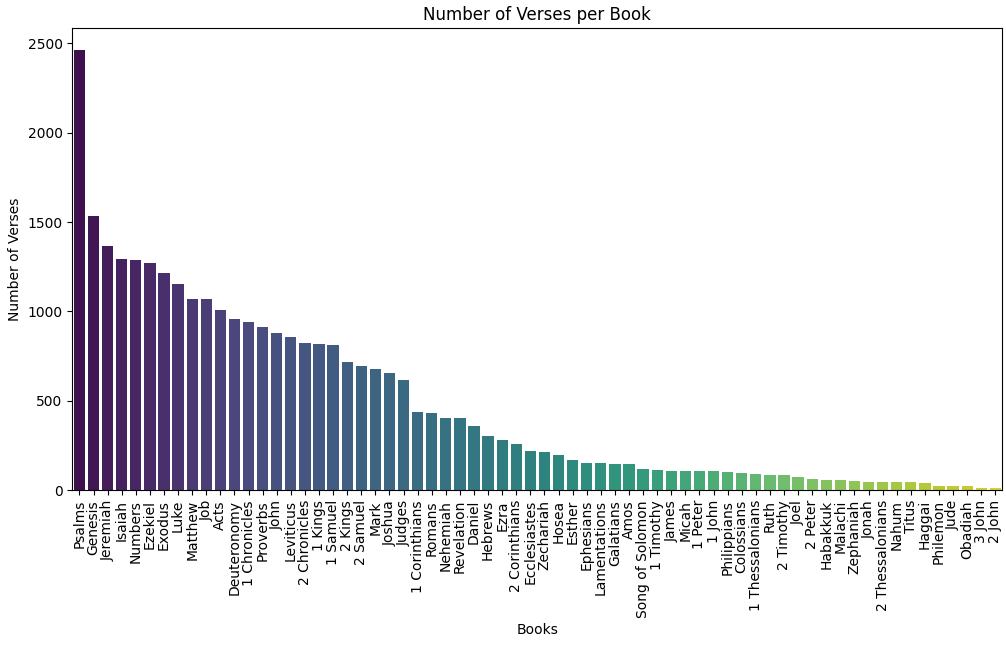
This phase involved exploring and analysing the dataset to gain insights into its structure, quality, content, and possible patterns. Key activities in this phase included:

* Analysing verses in the dataset.

*Figure 2: Range, Variance and Std from the dataset*

This is how bible verses are distributed in this dataset, having a range of 175 verses and an std of 14.07 of the mean.

* Examining the textual characteristics of Bible verses (e.g., length, common words).
  + Length:

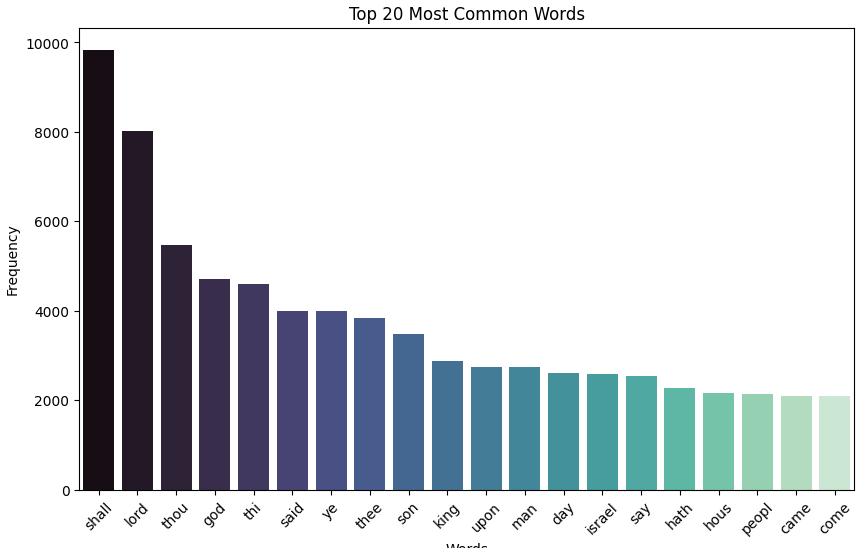
*Figure 3: Bar chart of number of verses per Bible book*

The book of Psalms being the one with more bible verses makes sense because it’s a book that expresses all possible emotions to God, which varies from time to time.

Something I could do with all this EDA information is to fit specifically problems to specific books, let’s say knowing that the book of psalms talks a lot about emotions I could train a model that uses more verse from the book of psalms when the topic given is Emotions, in that case I could take advantage of all this important information I’m getting from this dataset.

Another example would be. What did Jesus say about XYZ? So in that case I will go for Matthew, Mark, Luke, John and Revelation, to get exactly what were the words given by Jesus in that matter.

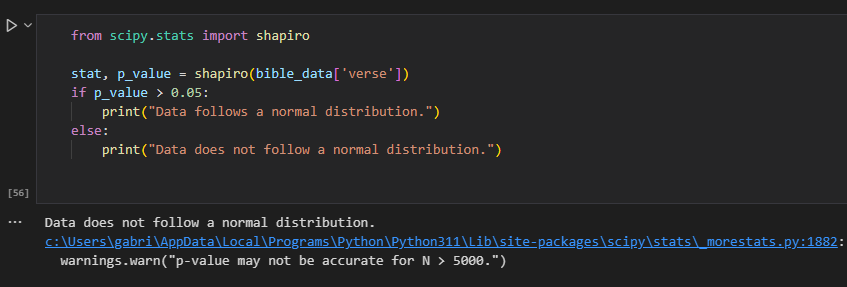
* + Common words:

*Figure 4: bar chart of the most common words before running stop words*

This common words can helps to create a classification of topics based on them.

* Identifying potential challenges, let's see a Normal distribution:

*Figure 5: Data distribution results*

**

As we can see, bible verses are not normally distributed, possibly creating us a challenge when classification models are in place.

Dataset Preparation

Raw text data, in this case, such as Bible verses, is unstructured, it needs to be processed in a way that the machine learning model can understand it work with it, and standardize it for further analysis.

To make sure the dataset was suitable for machine learning, these preprocessing techniques were applied to the Bible verses :

* Cleaning the data by removing stop words, tokenizing, and stemming the text to standardize it.

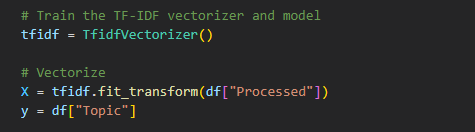
*Figure 6: Processing text function (tokenizing, stop words, stemming words)*

**Removing Stop Words**: getting rid of common words. "the," "a," "and," and "is" were eliminated because they did not add to the analysis, downgrading model performance.

**Tokenization**: Tokens are what you have when you split the text into smaller units. With this, we break down the Bible verses into individual words for further analysis.

**Stemming**: Reducing words to their root form is called stemming. For example, "running," "runner," and "ran" would have been reduced to "run." This helps a lot in standardizing words with similar or same meanings.

* One of the most common methods used in NLP is **TF-IDF** (Term Frequency-Inverse Document Frequency) which consists in converting textual data into numerical format as seen in picture 8. I trained all models except one with this vectorizer, because I wanted to be open to all kinds of possibilities during my model selection stage. As we will see later in this project, I found a great fit to solve this problem, but it was a harsh labour to do.

*Figure 7: TF-IDF Vectorizer at work*

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.

Now, after preprocessing, what I did is converted the textual data into a numerical format so that data can be added to a machine learning model. This process is known as well as **feature extraction.**

Web Scrapping Training Data

As part of this third CA, I wanted to improve the quality of my training data, and one of the best ways to do it is by increasing its size. Thanks to a Python web scraping strategy, I went from less than 20 rows to 758 rows.

**Steps:**

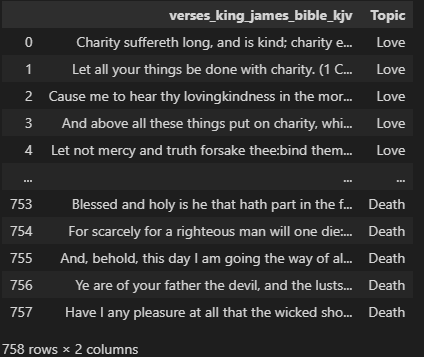
* + Found a respected website with bible verses grouped by topics.
  + In this case, I found DailyVerses.net.
  + If topics need to be changed, I will do it according to the website topics.
  + Webscrapp the verses and topic thanks to BeautifulSoup.
  + Coding the necessary code and Python implementations.
  + Save them into a df.
  + Train both the old Logistic Regression model from CA2 and the new models added in this CA
  + Compare models to find the best performance.

**Scrapping Function Explanation:**

* + Create the function and pass a desired topic
  + These topics will be entered automatically once I first define the function
  + There are 5 pages in each topic, so I'm taking all the verses in the 5 pages
  + Save the scraped verses into an empty list
  + Iterate through the main page to find the topic passed
  + Once the topic is found, get the "verses\_king\_james\_bible\_kjv" verses in the HTML
  + Append the verse and the book reference of the bible text at the end, very important.
  + Saved all to a df

Let's see a picture of how our training dataset looks like

*Figure 8: Train dataset overview*



The main goal with this training dataset is to train a model that would predict more accurately this kind of prediction task, then, once I find the model, use that very same model to pass it to our test set, which will be the original *bible\_data* dataset.

Machine Learning Implementation

**Training the Model**: The machine learning models were trained on the TF-IDF feature vectors, using the labeled topics (df['Topic']) as the target variable.

**Model Evaluations**: After training our models, we need to make sure that the bible verse predicted corresponds to the given topic. In this case, we took advantage of the widely common ML metrics such as Precision, Recall, F1 score, and accuracy:

1. Model Training and Evaluation

**Training the Model**: The machine learning models were trained on the TF-IDF feature vectors, using the labeled topics (df['Topic']) as the target variable.

**Model Evaluations**: After training our models, we need to make sure that the bible verse predicted corresponds to the given topic. In this case, we took advantage of the widely common ML metrics such as Precision, Recall, F1 score, and accuracy:

**Logistic Regression**.

Given a topic, the next step is to train a machine learning algorithm that can classify a Bible verse based on their content. At first instance, I chose Logistic Regression.

“ Logistic Regression is a powerful tool for text data, and very easy to understand how it works. In the NLP world is widely accepted, making Logistic Regression a go for text-related classification tasks” (Ganesan, 2020)

The logistic regression model calculates the probability of a verse belonging to a topic thanks to its features (TF-IDF scores) extracted from the verse in study.

**Random Forest**

“Random Forest has many advantages, for example, when comparing it with Decision Trees, the former builds an average of decision trees and gets the results from them. This approach reduces the problem of overfitting

Random Forests also handle imbalanced data well, making them a good option for our text classification task.” (CodeSignal, 2024)

This is a simple AI model that always comes into my mind when doing Machine Learning work. it's great reliability and the fact that it reduces the overfitting problem makes it my go-to model for this kind of task.

**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.

Later on, I added some parameter tuning and cross-validation techniques to strengthen even more this powerful model and compare with the plain version.

**Naïve Bayes**

“Both Naive Bayes and SVM classifiers are commonly used for text classification tasks.

Used with Python, Naive Bayes is a common used powerful machine learning algorithm when creating email spam filters, is particularly used for text classification tasks. The Naive Bayes algorithm relies on an assumption of conditional independence of features given when a class is given, which is actually a good approximation to real-world phenomena.” (Turing, 2025)

When studying the Google Advanced Data Analytics Certificate, I came across these Naïve Bayes theorems, wondering how powerful they would be in real-life scenarios, I decided to give it a chance in my look for a perfect model.

**WordVec + Logistic Regression**

This combination I found out looking for my perfect model on the internet. I do not know if it will work, but I love a challenge.

“Word2Vec for text classification is common in large text datasets.

One approach might be to represent a document as a vector by taking the average of the Word2Vec embeddings of the words in the document. Then, a classifier, like my first fit for this project, the Logistic Regression, would take this vector as an input.

Word2Vec embeddings turn words into vectors

I will tune these Word2Vec embeddings on my specific task by training a neural network to classify the text. In this approach, the Word2Vec embeddings are used as the input to the neural network, which is then trained to predict the class labels.

Word2Vec, combined with other machine learning techniques like cross validations, in this case I used a 3-fold cross validation, is a powerful tool for text classification”(Otten N.V, 2023)

Like I said in this one, I will get rid of the TF-IDF vectorizer and try this new one, Word2Vec with my first pic ever when I started this AI Bible App journey, the Logistic Regression. Keep reading to see the results.

**LSTM Model**

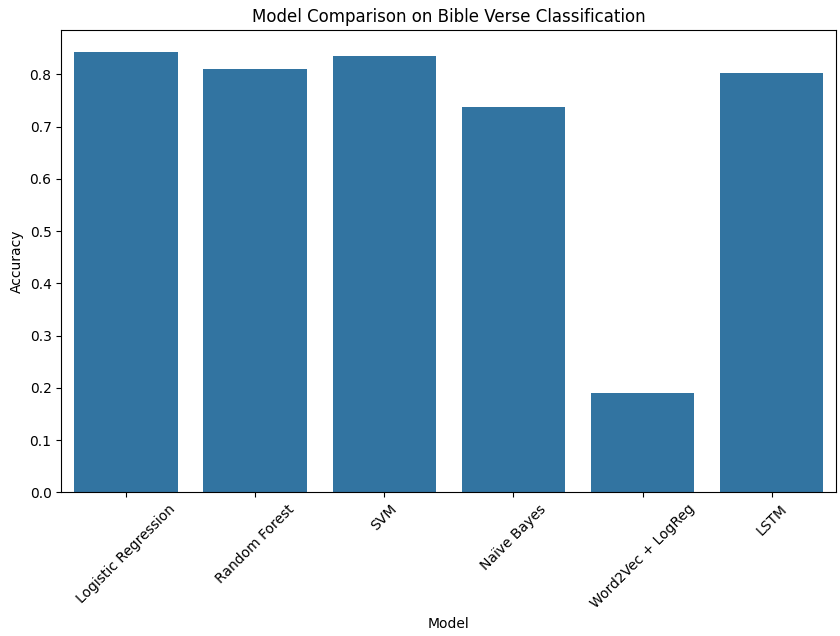
Working in prediction markets (Cryptocurrencies), I came across this powerful model. It is a great model for **patterns**, and this is exactly what I am looking for right now.

“A piece of text is a sequence of words, which might have dependencies between them. To learn and use long-term dependencies to classify sequence data, use an LSTM neural network.

To input text to an LSTM network, first, I need to convert the text data into numeric sequences. I can achieve this using a word encoding that maps documents to sequences of numeric indices like my TF-IDF vectorizer. For better results, also best practice is to include a word embedding layer in the network. Now, these embeddings capture semantic details of the words, which means that words with similar meanings have similar vectors, helping in building relationships. They also model relationships between words through vector arithmetic. For example, the relationship "Rome is to Italy as Paris is to France" would sound like: Italy - Rome + Paris = France.” (MathWorks, 2025)

An image is worth a thousand words, so let’s have a look at a small bar chart comparing all these models:

But before seeing that image, 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 get 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.

*Figure 9: Model Accuracy Comparisons*

2. Model Evaluation and Conclusion: Accuracy Scores

**Logistic Regression**:  0.84, not bad results, that is why this was my first choice in the previous version. After some retraining attempts sometimes it went up as far as 0.85

**Random Forest**: 0.81, RF normally works great on text classification, and in this case, it made it to the top three performers. After some retraining attempts sometimes performed the best of all.

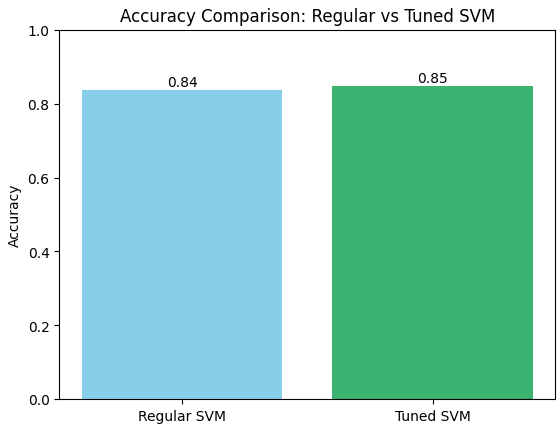
**Naïve Bayes**: 0.74, I will not say poor performance, because it was the worst among the others. Naïve Bayes normally works better in smaller datasets.

**Fined- Tuned Models:**

**Word2Vec + Logistic Regression:** 0.19, the most difficult to code the worst performer of all. Tried to tuned it but seem like Word2Vec is not working for this dataset at least.

**LSTM Model**: 0.80, One of my favourites sophisticated modes, I came across this model multiple times in multiple occasions when building personals projects. Yet I have not found any sophisticated model that performed better than a simpler one in simper tasks. With that being said, the SVM model will make it to the last round.

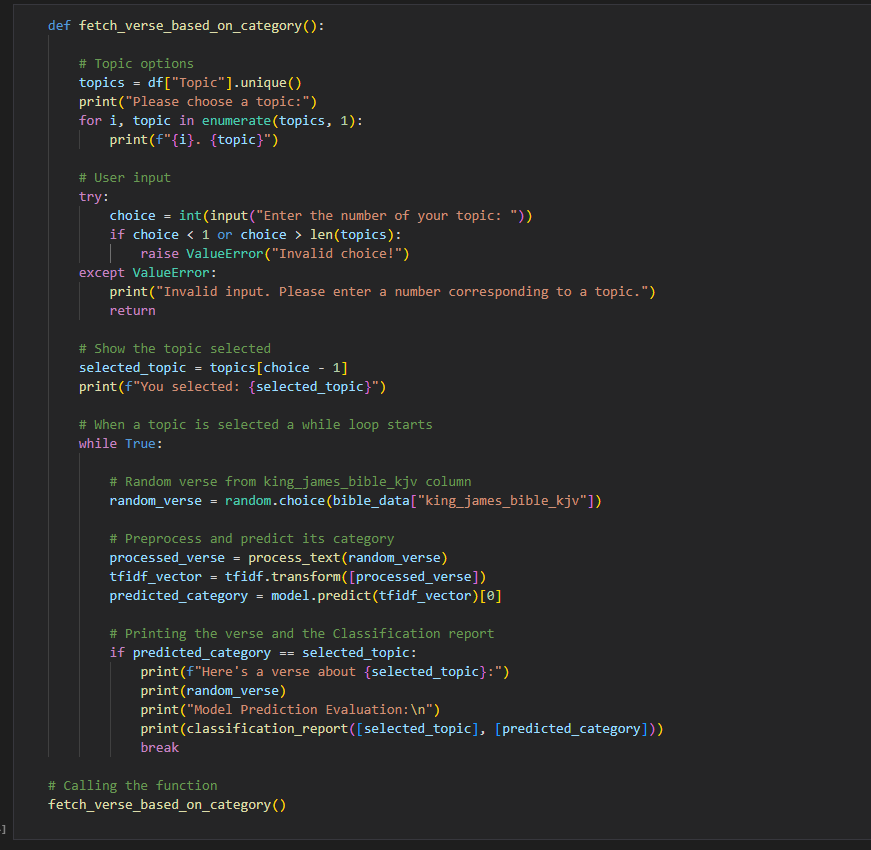
**SVM and Tuned**: Regular: 0.85 and sometimes 0.84. Tuned: 0.85, by a small difference, this is the best model performer I found until this moment. After tuning it with cross-validation techniques, I found out that it even performed better. This one steadily performs the same after multiple retraining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

*Figure 10: Tuned SVM vs Regular SVM*

3. Deployment: Bible Verse Recommendations System

When tuning the SVM, I found out that the closer I tuned to the original one, the better results I had. In this case, Tuned SVM still performs better than Regular SVM.

After finding my perfect fit, I decided to enter the Designing phase of the Roadmap and implemented an user interface so people can interact with the program:

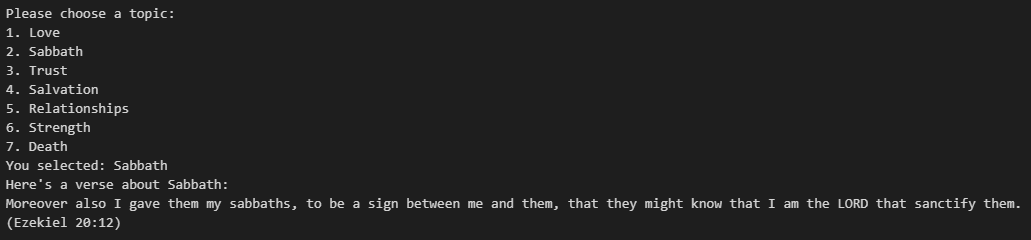
***Figure 11: User interface logic with machine learning implementation*

Let me explain the workflow:

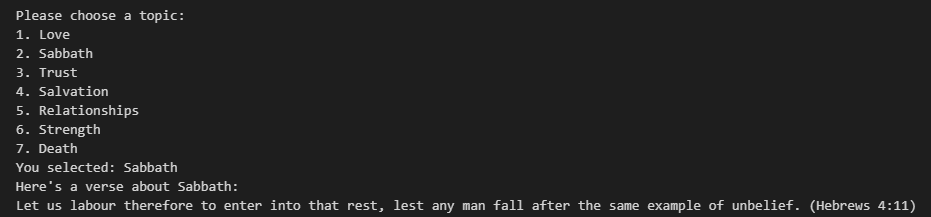
1. **User Topic Selection**: By just selecting a number the user chooses a topic of interest from a list of categories (Love, Salvation, Trust, Guidance, Relationships, and Health).
2. **Random Verse Selection**: Once a topic is selected, the code looks for a random verse from the Bible dataset.
3. **Prediction**: When the algorithm finally finds what it thinks is an appropriate verse it prints it in the screen.

Now let’s see if this app is working properly, let’s say I want to predict a verse about the Sabbath topic, let’s see two examples:

*Figure 12: First prediction about the Sabbath Topic*

**

*Figure 13: Second prediction about the Sabbath Topic*

**

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.

Challenges

While it's true that this project achieved its primary goals, we have several challenges for further improvement:

1. **Data Imbalance**: Some topics such as "Love" and "Salvation," have more verses than others, which can lead to an imbalance in the dataset. This could potentially cause the algorithm to be inaccurate. I am thinking about oversampling, undersampling, or using weighted loss functions that could help fix this issue.
2. **Model Accuracy**: It is great that I did not enter into an infinite loop trying to find a bible verse but we saw the poor performance on this algorithm. Future work will include exploring more advanced models can hopefully improve Bible verse predictions. I will take advantage of GridsearchCV or any custom ways to make this algorithm work.
3. **Designing User Experience**: Of course that is not our end design, we are into the MVP phase, so creating like a web app using an easy Python framework for example Flask or FastAPI is going to be something that I will consider in the future for this project in order to have a great future experience in our app.
4. **Multilingual Support**: My mother language is Spanish, I love reading my bible in Spanish, next time we could potentially find another Kaggle dataset with Spanish bible verses and join it with our current one, at the moment it is not nothing in our top priorities.

**Results and Analysis… and next steps**

* + Out of our three fine-tuned modes **(LSTM, Word2Vec + Logistic Regression**, and **Tuned SVM**), the latter performed better than the previous ones.
  + It works great when at least the root word of the given topic is found in the text but when I need a verse that does not have the root word in the verse, the results hallucinates a bit.
  + One cool feature we can add to this dataset: it would be better off if instead of the whole bible I load the most common bible verses from the Scriptures (In this case, I would need to change the whole bible\_data dataset), skipping out of context verses. (e.g: When trying to predict a verse about the Sabbath topic, not because the word "day" is in the verse, does not necessarily mean that it is a text about the sabbath).
  + Now with the growth of AI, specially in LLM, this app could have been done with a few lines of code, a simply API call to OpenAI databases, (Which is not an open source product, but in terms of adoption it is the most accepted one, if an open source needed I could use Deepseek) Leveraging ChatGPT (One of the most if not the most reliables data to text model) accuracy in text prediction.
  + Overall, the building of this predicting bible verses project has been successful, from data overview to model evaluation. Having an accuracy of 0.85 helps a lot in terms of text prediction, I have to say that it surprises me how it is able to find bible verses related to the given topic. Still, there is room for improvement.
  + In the future, I could take advantage of my EDA to create a better solution for specific topics. Let's say an user needs some words of wisdom, I would fit a model that reads more data from the Proverbs book than any other bible book.
  + Without letting down the Data Analytics side of this study, something that I learnt along the process is that the simpler the better. My simplest models gave me the best results.

Conclusion

This is a very interesting topic which I am grateful to work on, but there are a lot of things to do. In order to have a great functional AI Bible app I must fit the perfect algorithm for it, anything comes after it.

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

Chumbar, S. (2023). The CRISP-DM Process: A Comprehensive Guide. [online] Medium. Available at: <https://medium.com/@shawn.chumbar/the-crisp-dm-process-a-comprehensive-guide-4d893aecb151>.

GeeksforGeeks (2024). 5 Simple Ways to Tokenize Text in Python. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/5-simple-ways-to-tokenize-text-in-python/ [Accessed 26 Dec. 2024].

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

GeeksforGeeks. (2021). Understanding TF-IDF (Term Frequency-Inverse Document Frequency). [online] Available at: <https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/>.

Kandola, A. (2023). Neuro-linguistic programming (NLP): Does it work? [online] www.medicalnewstoday.com. Available at: <https://www.medicalnewstoday.com/articles/320368#techniques>.

Lawton, G. (2022). What is Logistic Regression? - Definition from SearchBusinessAnalytics. [online] SearchBusinessAnalytics. Available at: <https://www.techtarget.com/searchbusinessanalytics/definition/logistic-regression>.

Otten, N.V. (2023). 9 Ways To Use Text Normalization Techniques In NLP With Python. [online] Medium. Available at: https://medium.com/@neri.vvo/9-ways-to-use-text-normalization-techniques-in-nlp-with-python-848f32480c98 [Accessed 26 Dec. 2024].

Pykes, K. (2023). Stemming and Lemmatization in Python. [online] www.datacamp.com. Available at: <https://www.datacamp.com/tutorial/stemming-lemmatization-python>.

Simha, A. (2021). Understanding TF-IDF for Machine Learning. [online] Capital One. Available at: <https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/>.

Walker, S. (2023). F-Score: What are Accuracy, Precision, Recall, and F1 Score? — Klu. [online] klu.ai. Available at: <https://klu.ai/glossary/accuracy-precision-recall-f1>.

Wikipedia Contributors (2019). tf–idf. [online] Wikipedia. Available at: <https://en.wikipedia.org/wiki/Tf%E2%80%93idf>.

Qudirah (2023). Sentiment Analysis Project Using TextBlob. [online] Medium. Available at: <https://medium.com/@qudrohbidemi/sentiment-analysis-project-using-textblob-216d3fe119fc>.

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>.

Breuss, M. (n.d.). Beautiful Soup: Build a Web Scraper With Python – Real Python. [online] realpython.com. Available at: <https://realpython.com/beautiful-soup-web-scraper-python/>.

Brownlee, J. (2019). Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/>.

Code with Josh (2025). LSTM Time Series Forecasting with TensorFlow & Python – Step-by-Step Tutorial. [online] YouTube. Available at: https://www.youtube.com/watch?v=94PlBzgeq90 [Accessed 11 Apr. 2025].

GeeksForGeeks (2019). SVM Hyperparameter Tuning using GridSearchCV | ML. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/svm-hyperparameter-tuning-using-gridsearchcv-ml/>.

Radečić, D. (2024). How to Visualize Machine Learning Models: From Linear Regression to Neural Networks. [online] Datacamp.com. Available at: <https://www.datacamp.com/tutorial/visualize-machine-learning-models>.

www.datacamp.com. (n.d.). Python LSTM (Long Short-Term Memory Network) for Stock Predictions. [online] Available at: <https://www.datacamp.com/tutorial/lstm-python-stock-market>.

GitHub Link

<https://github.com/CCT-Dublin/ca1-capstone-project-proposal-Gabriel-studies>

Presentation Link

<https://drive.google.com/file/d/1CwAD7fbJYsmSBE8WZfcsolAHRrWfD8wM/view?usp=sharing>

Dataset Link

<https://drive.google.com/drive/folders/1FVvDtJyV0r1HZVmJw4LGV7P5cLcqc6hL?usp=sharing>