Logistic Regression and Text Classification as part of the Yeshiva University Computer Science Capstone Project

Thesis Submitted in Partial Fulfillment

of the Requirements

of the Jay and Jeanie Schottenstein Honors Program

Yeshiva College

Yeshiva University

May 2023

Jacob Silbiger

Mentor: Professor Dave Feltenberger, Computer Science

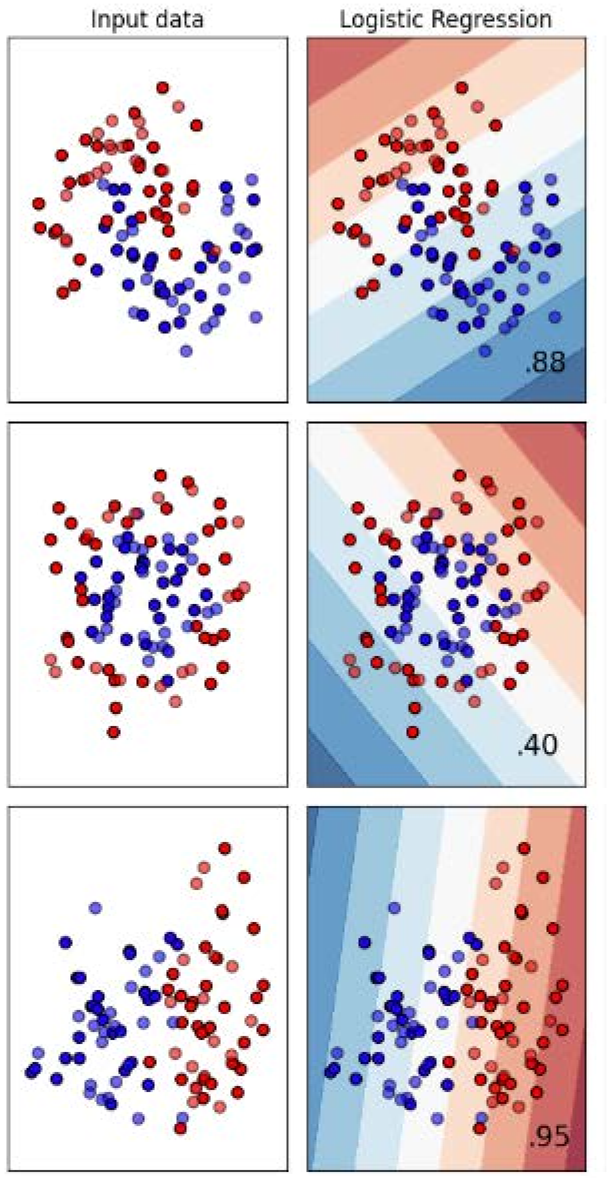
Introduction

Text classification is a machine-learning task of assigning labels to a given text. The texts can be categorized based on topics or the sentence's sentiment like this is a positive or negative sentence. In my capstone project, we were tasked with correctly labeling texts from the Sefaria dataset. The project aims to create a machine-learning model to predict a given Hebrew text's correct topic or topics. One of my tasks in the project was building logistic regression models for text classification. The results of these models became the initial performance bar to measure the results of other models created in the project. This paper aims to explain what logistic regression is, how to use it, and how it applies to our project.

Logistic Regression

Logistic regression is a machine-learning model used for classification tasks. Classification is when the machine learning model assigns the item to a given category or multiple categories. The model returns a probability of what classification it believes the item belongs to.

Logistic regression is sometimes used as the benchmark model as it is one of the simplest to create. It is also more straightforward than most neural networks. Because we know the coefficients ($w_1 ... w_n$), we can easily interpret the results from a logistic regression model. Some downsides are that the model can sometimes overfit with a more complex dataset. This means that the model learns to fit the given data but can not apply it to new data. There are ways to help minimize this issue called L1 and L2 regularization. Also, the data must be linearly separated, as the model can not learn complex relationships. Logistic regression has a linear decision boundary, meaning the cutoff between $y = 0$ and $y = 1$, is a straight line. As seen in this image, the first two rows are non-linearly separated data. There cannot be a line such that it can divide the data linearly into categories. Therefore, the accuracy of the logistics regression model is low. While the last row is linearly separated data, and the accuracy is 0.95



(Grobler et al.)

Logistic regression models can also be sensitive to outlying data, which can skew the model. Fortunately, for text classification, logistic regression works reasonably well (Grover).

As mentioned, other machine learning models can be used for text classification. These models are more complex than logistic regression and can sometimes lead to better results. Nevertheless, the simplicity and efficiency of logistic regression make it a great benchmark tool before starting the more advanced methods and models (Grover).

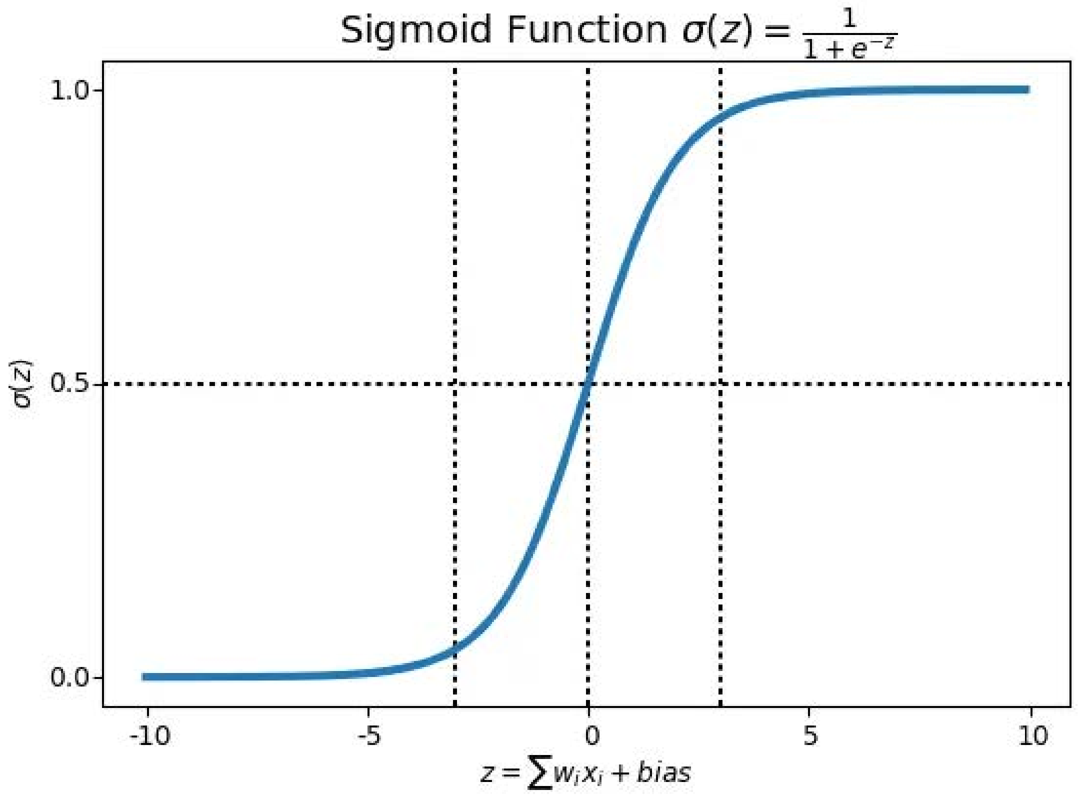
How Logistic Regression Works

Logistic regression is similar to linear regression using independent and dependent variables. Logistic regression requires the linear relationship between the independent variable X and the dependent variable, the known outcome Y. The formula for a linear relationship is $y = b + w_1 * X_1 + w_2 * X_2 + ... + w_n + X_n$. Where X_i represents the data variables, y is the predicted outcome, and b is the y-intercept when we set all X values to 0, $w_1 ... w_n$ are the coefficient, also known as the weights. The goal of regression models is to change the coefficients, $w_1 ... w_n$, to minimize the difference between the actual Y value and the predicted y value from the formula for all Y values. This is known as the cost function (Gupta)

Sigmoid Function

Logistic regression can only predict a probability between 0 and 1 because it uses the "Sigmoid Function." This is the probability that the object is in a given class, with a probability greater than 0.5 meaning that the object is in the given class and less than 0.5 being not in the class. Similarly, if the result is closer to 0 or 1, then the model is more confident that the item is in that particular group. The sigmoid function is used in many machine learning models where one tries to predict between two classes. The sigmoid function formula is $\sigma(z) = \frac{1}{1+e^{-(z)}}$, where z represents the formula on which we want to apply the sigmoid function. So using the regression formula to the sigmoid formula $z = b + w_1 * X_1 + w_2 * X_2 + ... + w_n + X_n$,

we arrive at the function $\sigma(z) = \frac{1}{1+e^{-(b + w_1 * X_1 + w_2 * X_2 + ... + w_n + X_n)}}$.



For logistic regression, we need a cost function for each label, 0 and 1. The cost function is $- log(\sigma(z))\ if\ y = 1$ and $- log(1 - \sigma(z))\ if\ y = 0$. Which can be combined into one formula $J = - \sum ((y_i * log(\sigma(z_i)) + (1 - y_i) * log(1 - \sigma(z_i))))$ for all y values. The goal of the model is to minimize J, this achieved through applying the gradient descent algorithm on each weight. The formula for gradient descent is: $\theta_{new\,j} = \theta_{old\,j} - a\,\frac{\partial J(\theta)}{\partial(\theta_j)}$ where θ_j represents each weight ($w_1 ... w_n$). a is a constant and is known as the learning rate (Pant).

Understanding the Coefficients

Since logistic regression aims only to predict values between 0 and 1. the goal of our equation is $\sigma(z) = log\left(\frac{P\{Y=1\}}{P\{Y=0\}}\right)$, or the log of probability that $y = 1$ is divided by the probability that $y = 0$. This is known as $logit(p)$ and also log-odds. Therefore, when we change w_i, we are changing the probability of $log\left(\frac{P\{Y=1\}}{P\{Y=0\}}\right)$. When we increase w_i, we increase the probability of $Y = 1$, when we decrease w_i, we decrease the probability of $Y = 1$, increasing the probability of $Y = 0$ (Kisselev).

How to Use logistic regression in Python – Sklearn

A simple and efficient way to use logistic regression is using the Scikit Learn library. To build a model, one imports the logistic regression tool and then calls the logistic regression method LogisticRegression(). The logistic regression method allows us to pass in parameters, from helping with overfitting to changing the starting coefficients. In any case, the base model with no parameters works well.

To train the model, one passes in the data by calling LogisticRegressionModel.fit(X,y), where X is the data and y is the results. So now we have a trained model, and we can use the model to make predictions on new data. We can also call predict(X) and pass in new data to predict the results of unseen data. The model also had a method called predict_proba(X), which allows us to see the probabilities for each class for any X we pass in. There is also a method called score(X,y) which returns the accuracy of the given test data and labels. Here is a demo code from the Scikit Learn website (>>> indicates a line of code, no arrows indicate output to the screen). In this case, the iris data is set to X and y, then it makes a logistic regression model and calls fit passing in the X and y data. The code then calls predict, predict_proba on the first two pieces of X data, and score on all of the X data to show the different possible ways to view the model results:

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.linear_model import LogisticRegression
>>> X, y = load_iris(return_X_y=True)
>>> clf = LogisticRegression(random_state=0).fit(X, y)
>>> clf.predict(X[:2, :])
array([0, 0])
>>> clf.predict_proba(X[:2, :])
array([[9.8...e-01, 1.8...e-02, 1.4...e-08],
    [9.7...e-01, 2.8...e-02, ...e-08]])
>>> clf.score(X, y)
0.97…
```

As explained, logistic regression allows one to look at the coefficients ($w_1 ... w_n$) and to see what is important for the model. The Scikit logistic regression model has a built-in method called call coef_ which allows one to view the model’s coefficients (*Sklearn.linear_model.logisticregression*).

Logistic Regression and Text Classification

To use the logistic regression model, one needs both the x and y values to be numbers or a list of numbers. To use logistic regression for text classification, one needs a way to turn texts and the classification topic into numbers that make sense and that the model can understand. For the topics, one can easily say if the text is in the topic, then return 1; if the text is not, then return 0.

For the texts, there are a couple of ways to turn the text into usable numbers. The first way is called Bag of Words. This way is the simplest and still produces good results. The Bag of Words is just a list of all other words seen in the data, and for each text, observe how many times a word appears from the vocabulary list. Here is an example from a Medium article:

> Let’s take an example to understand this concept in depth.
>
> “It was the best of times”
>
> “It was the worst of times”
>
> “It was the age of wisdom”
>
> “It was the age of foolishness”
>
> We treat each sentence as a separate document and we make a list of all words from all the four documents excluding the punctuation. We get,
>
> ‘It’, ‘was’, ‘the’, ‘best’, ‘of’, ‘times’, ‘worst’, ‘age’, ‘wisdom’, ‘foolishness’[1]
>
> The next step is the create vectors. Vectors convert text that can be used by the machine learning algorithm.

[1] This list of words is called a vocabulary and the order of this list stays the same. The vectors based on this list can be applied to all documents. For example, the word “it” will always represent the first item in the vector, for any document using this vocabulary.

We take the first document — “It was the best of times” and we check the frequency of words from the 10 unique words.

“it” = 1

“was” = 1

“the” = 1

“best” = 1

“of” = 1

“times” = 1

“worst” = 0

“age” = 0

“wisdom” = 0

“foolishness” = 0

Rest of the documents will be:

“It was the best of times” = [1, 1, 1, 1, 1, 1, 0, 0, 0, 0]

“It was the worst of times” = [1, 1, 1, 0, 1, 1, 1, 0, 0, 0]

“It was the age of wisdom” = [1, 1, 1, 0, 1, 0, 0, 1, 1, 0]

“It was the age of foolishness” = [1, 1, 1, 0, 1, 0, 0, 1, 0, 1]

(D'Souza)

Now that we have turned the text into a numerical representation, we can pass this data into the logistic regression model (D'Souza).

Like the logistic regression model, Scikit also has a tool for creating a bag of words called sklearn.feature_extraction.text.CountVectorizer. This tool also allows us to do preprocessing on the data, for example, set all letters to lowercase or remove all non-alphanumeric characters. Here is the example code from the Scikit Learn website. First, we take a dataset. In this case, “corpus” is the list of sentences. Then we call the CountVectorizer() to create a bag of words vectorizer method. Then we make the bag of words for the texts by calling fit_transform(corpus), passing in the corpus data, and saving the output to a variable called X. Next, one can use get_feature_names_out() can be used to show the words in the vocabulary. The last line shows the texts saved as these new frequency lists

```
>>> from sklearn.feature_extraction.text import CountVectorizer
>>> corpus = [
... ‘This is the first document.’,
... ‘This document is the second document.’,
... ‘And this is the third one.’,
... ‘Is this the first document?’,
... ]
>>> vectorizer = CountVectorizer()
>>> X = vectorizer.fit_transform(corpus)
>>> vectorizer.get_feature_names_out()
array([‘and’, ‘document’, ‘first’, ‘is ‘, ‘one’, ‘second’, ‘the’, ‘third’,
    ‘this’], ...)
>>> print(X.toarray())
[[0 1 1 1 0 0 1 0 1]
 [0 2 0 1 0 1 1 0 1]
 [1 0 0 1 1 0 1 1 1]
 [0 1 1 1 0 0 1 0 1]]
```

(*Sklearn.feature_extraction.text.CountVectorizer*)

Another way to fit the data is using TF-IDF. TF-IDF stands for term frequency-inverse document frequency. TF-IDF evaluates how important a word is to a document compared to all other documents. The significance increases proportionally based on the number of times a word appears in a given document but is balanced by the frequency of that word in all of the documents. TF-IDF scores can be created by calling sklearn.feature_extraction.text.TfidfVectorizer (D'Souza)(*Sklearn.feature_extraction.text.TfidfVectorizer*).

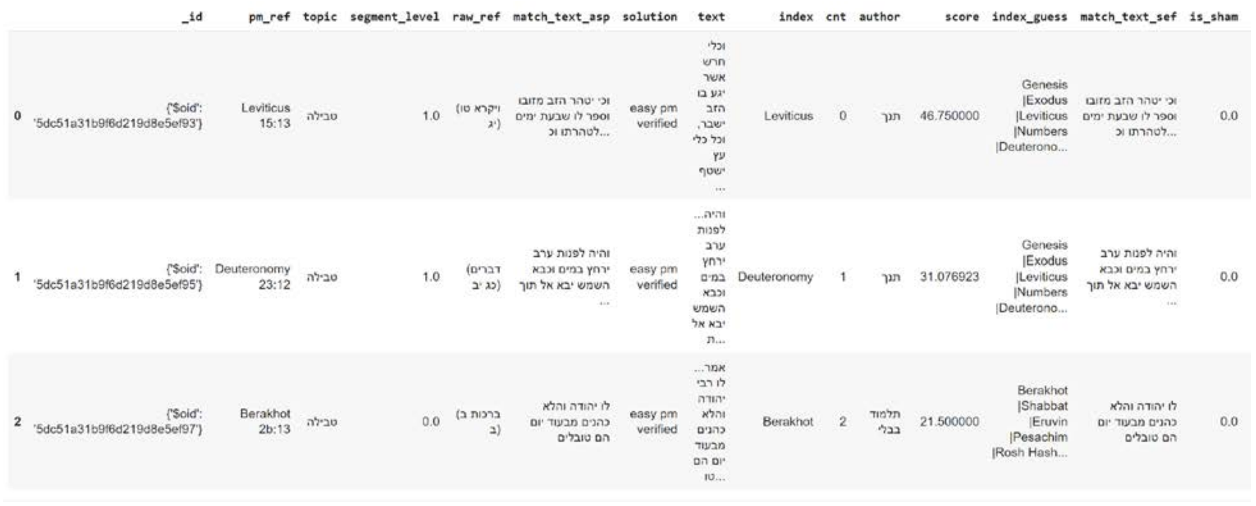
Computer Science Capstone Project

The Sefaria website is a library of Jewish texts and their interconnections in Hebrew and other translations. Sefaria, a non-profit organization, has been one of the leaders in online Jewish texts and education over the past decade. Utilizing their library of several hundred million words of Jewish texts, they have begun to develop a catalog of topics. Anyone looking to learn more can search the topics catalog to find sources from various works of their online library, all in one place (*About Sefaria*).

Until now, Sefaria's topic text taggings have been based on (and limited to) entries in Aspaklaria, an encyclopedia of Jewish thought. They are looking for ways to expand their tagging to the rest of the texts in their online library. To help their efforts, we have made several machine learning models that can learn from previously labeled texts to tag more unlabeled texts in the Sefaria library. Our basic approach is to make a binary classifier for each topic, where, based on a confidence threshold, the model will label a text as belonging or not belonging to a topic. As mentioned, my role was to create the logistic regression models to be used as the benchmark model to compare the other models against.

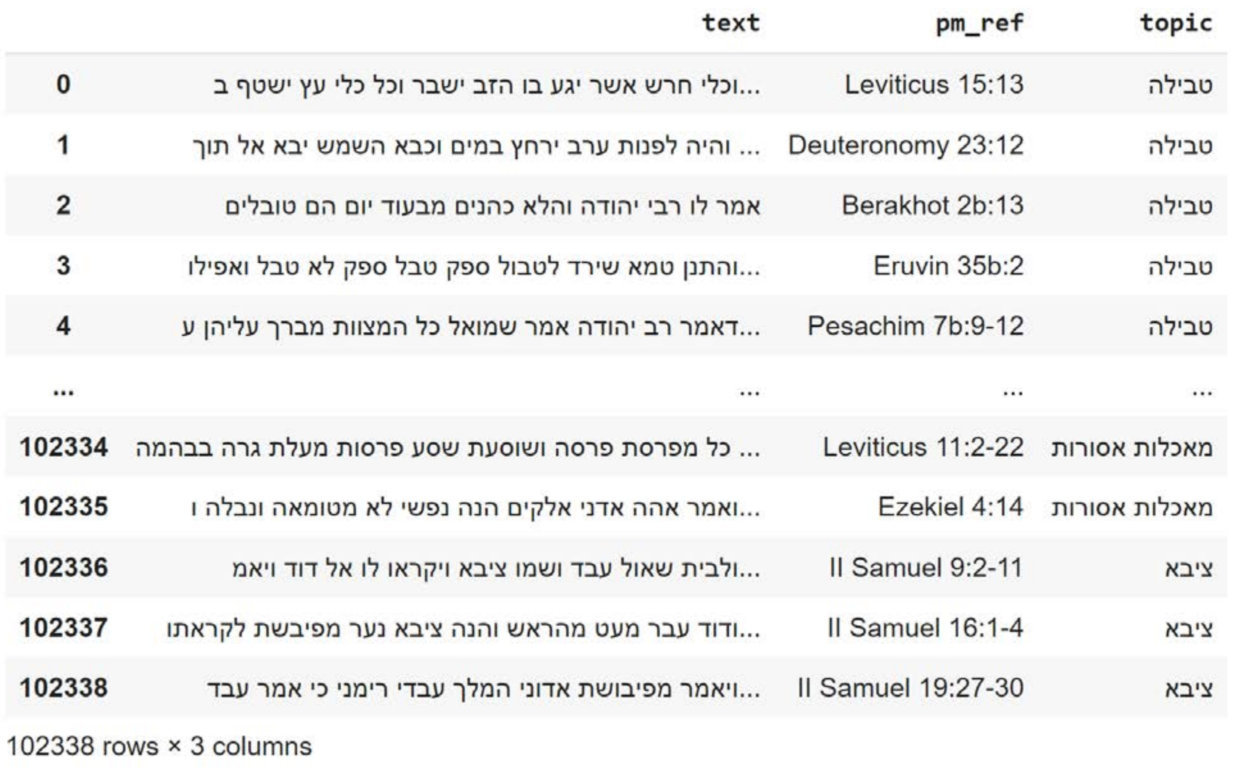
Sefaria Dataset

The data given to us by Sefaria was from an encyclopedia named Aspaklaria. The dataset contained the text, the reference location, the topic, and other fields relating to the text or used for Sefaria's internal use. Here is a sample of the data:



	_id	pm_ref	topic	segment_level	raw_ref	match_text_asp	solution	text	index	cnt	author	score	index_guess	match_text_sef	is_sham
0	{'$oid': '5dc51a31b9f6d219d8e5ef93'}	Leviticus 15:13	טבילה	1.0	(ויקרא טו יג)	וכי יטהר הזב מזובו וספר לו שבעת ימים לטהרתו וכ...	easy pm verified	וכלי חרש אשר יגע בו הזב ישבר, וכל כלי עץ ישטף ...	Leviticus	0	תנך	46.750000	Genesis |Exodus |Leviticus |Numbers |Deuterono...	וכי יטהר הזב מזובו וספר לו שבעת ימים לטהרתו וכ...	0.0
1	{'$oid': '5dc51a31b9f6d219d8e5ef95'}	Deuteronomy 23:12	טבילה	1.0	(דברים כג יב)	והיה לפנות ערב ירחץ במים וכבא השמש יבא אל תוך ...	easy pm verified	והיה... לפנות ערב ירחץ במים וכבא השמש יבא אל ת...	Deuteronomy	1	תנך	31.076923	Genesis |Exodus |Leviticus |Numbers |Deuterono...	והיה לפנות ערב ירחץ במים וכבא השמש יבא אל תוך ...	0.0
2	{'$oid': '5dc51a31b9f6d219d8e5ef97'}	Berakhot 2b:13	טבילה	0.0	(ברכות ב ב)	לו יהודה והלא כהנים מבעוד יום הם טובלים	easy pm verified	אמר... לו רבי יהודה והלא כהנים מבעוד יום הם טו...	Berakhot	2	תלמוד בבלי	21.500000	Berakhot |Shabbat |Eruvin |Pesachim |Rosh Hash...	לו יהודה והלא כהנים מבעוד יום הם טובלים	0.0

From here, we extracted the texts, topics, and reference locations to create a smaller dataset that was easier to use. We also cleaned the texts of any punctuation or non-Hebrew characters and removed any duplicates or missing data.



	text	pm_ref	topic
0	וכלי חרש אשר יגע בו הזב ישבר וכל כלי עץ ישטף ב...	Leviticus 15:13	טבילה
1	והיה לפנות ערב ירחץ במים וכבא השמש יבא אל תוך ...	Deuteronomy 23:12	טבילה
2	אמר לו רבי יהודה והלא כהנים מבעוד יום הם טובלים	Berakhot 2b:13	טבילה
3	והתנן טמא שירד לטבול ספק טבל ספק לא טבל ואפילו...	Eruvin 35b:2	טבילה
4	דאמר רב יהודה אמר שמואל כל המצות מברך עליהן ע...	Pesachim 7b:9-12	טבילה
...	...	...	...
102334	כל מפרסת פרסה ושוסעת שסע פרסות מעלת גרה בבהמה ...	Leviticus 11:2-22	מאכלות אסורות
102335	ואמר אהה אדני אלקים הנה נפשי לא מטומאה ונבלה ו...	Ezekiel 4:14	מאכלות אסורות
102336	ולבית שאול עבד ושמו ציבא ויקראו לו אל דוד ויאמ...	II Samuel 9:2-11	ציבא
102337	ודוד עבר מעט מהראש והנה ציבא נער מפיבשת לקראתו...	II Samuel 16:1-4	ציבא
102338	ויאמר מפיבושת אדוני המלך עבדי רימני כי אמר עבד...	II Samuel 19:27-30	ציבא

102338 rows × 3 columns

In the end, there were 102,338 rows of data.

For the logistic regression benchmark, we created models for the top five most prevalent topics' תשובה', 'תפלה', 'תורה', 'למוד', and 'ישראל' with 998, 894, 837, 711, and 695 texts, respectively.

Turning Sefaria Data Into Readable Data for the Model

For each topic, I created its own binary classification model. Either the text is in the topic or is not. For each topic, I created a dataset from the larger dataset in which half of the texts were from the topic, and half were not. All texts from the given topic were labeled 1, and the rest were labeled 0. I then created a bag of words for the new dataset and set each frequency list as the representation of the texts. I also removed certain words called stopwords, which are words in the text but do not add much meaning to the word. In English, this can be the words like "a" or "the."

The texts were then shuffled and split into two datasets, one to train the model and one to test how well the model performed. The model does not see this second dataset during training, and the results from the test dataset indicate how well the model actually learned the data.

The first dataset was used to train a basic logistic regression model, with the bag of word frequency list as the X and the topic, represented by 0 and 1, as the Y.

Results

Precision, Recall, F1-score, Accuracy

There are four main ways to interpret how well a classifier performed, in this case, the logistic regression model. The first way is the simplest, which is accuracy. This measures the number of correct predictions/ all predictions made. This gives us an overview of the model but does not explain why it performed as well as it did. For example, if there was a dataset with ten texts, three texts are labeled 0, and 7 are labeled 1, $Y_{true} = [0, 0, 0, 1, 1, 1, 1, 1, 1, 1]$. Let us say our model predicts every text as label 1, $y_{results} = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$. Then our accuracy will be $7\ (correct\ labels)/10\ (all\ labels)$, which is 70% accuracy. However, this is still not a good model as it failed to predict any labels where $y = 0$. Therefore, we use Precision, Recall, and F1-score to help explain our results.

TP, TN, FP, FN

When dealing with Precision, Recall, and F1-score, we use the terms Positive and Negative to describe the data. Positive is when the data is in the class (1), and Negative is when the data is not in the class (0). From here, we have four possibilities: True Positive, True Negative, False Positive, and False Negative.

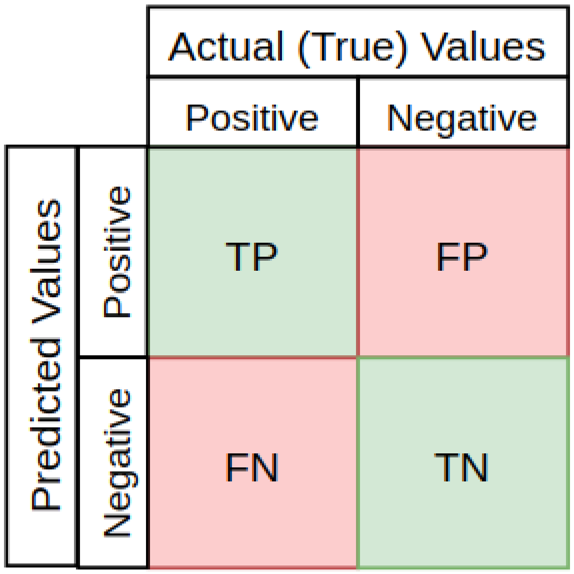
True Positive (TP) is when the object is in the class, and the classifier correctly classified it as in the class (Y true =1, Y prod =1).

False Positive (FP) is when the object is not in the class, and the classifier classifies it incorrectly as in the class (Y true =0, Y prod =1).

True Negative (TN) is when the object is not in the class, and the classifier correctly classified it as not in the class (Y true =0, Y prod =0).

False Negative (FN) is when the object is not in the class, and the classifier classifies it incorrectly as in the class (Y true =0, Y prod =1).

Here is an image representation of the four possibilities:

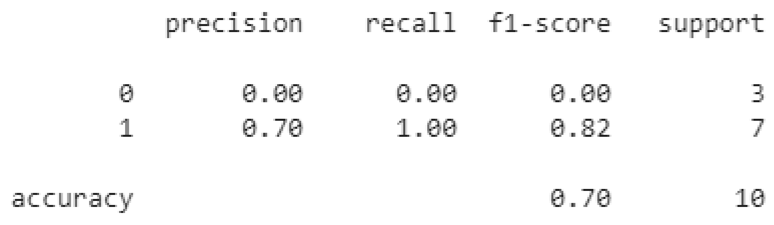


		Actual (True) Values	
		Positive	Negative
Predicted Values	Positive	TP	FP
	Negative	FN	TN

We can now understand Precision, Recall, and F1-score with these terms. Precision measures how many positive predictions the classifier makes are correct. The formula for

Precision is: $TP/(TP + FP)$. Recall measures how many positive cases the classifier labeled correctly and using the formula: $TP/(TP + FN)$. F1-score combines these two scores into one metric. F1-score is: $(2 * Precision * Recall)/ (Precision + Recall)$. The F1-score requires both precision and recall to be high for the F1-score to be high as well (Kanstrén).

Here is an image of the precision, recall, F1-score, and accuracy of the example above:

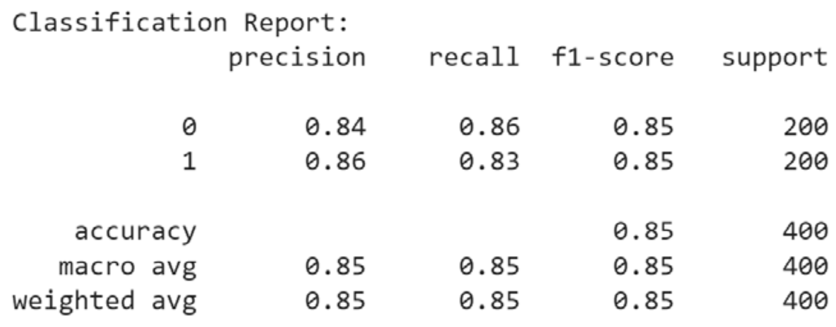


```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.70      1.00      0.82         7

    accuracy                           0.70        10
```

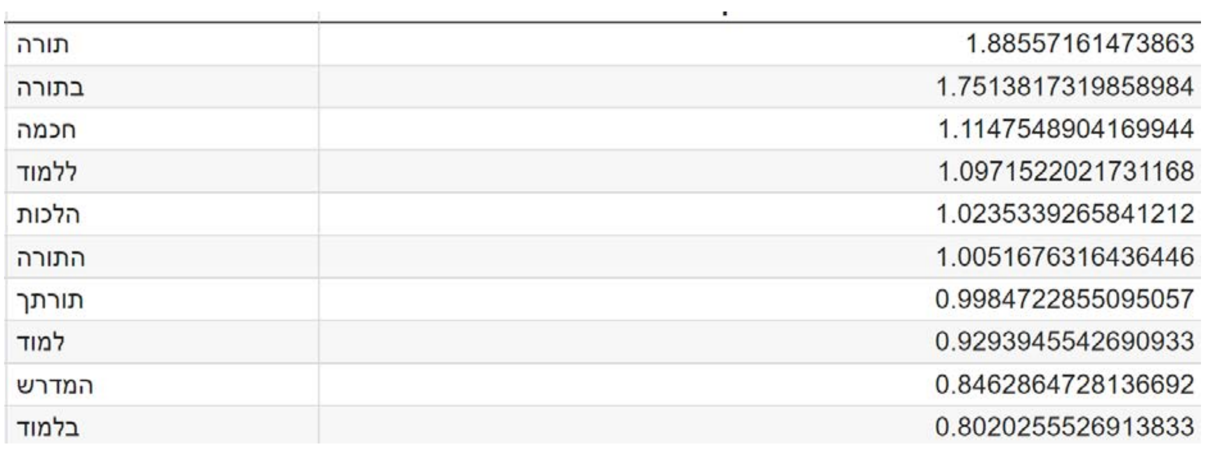
With Precision, Recall, and F1-score, we can now look at the results from the logistic regression models. For each topic, its model performed between 80% and 90% accuracy as well as Precision, Recall, and F1-scores, with some scores in the high 90s. As explained, logistic regression allows us to view the coefficients and see which words are the most important for the model. In addition, this will enable us to see how well the model learned the language and if it understands what to look for.

Here are the five models' results with the top 10 most important words.

למוד



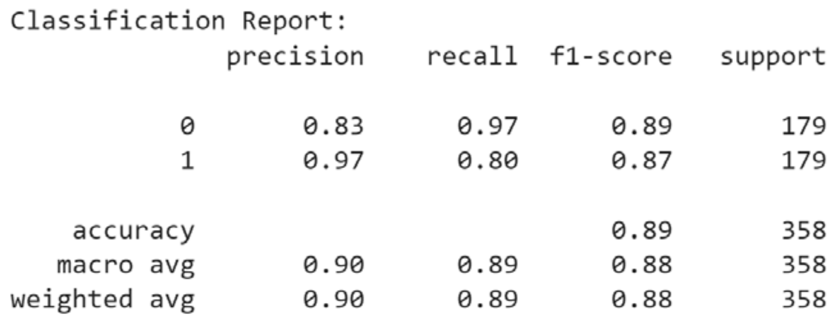
```
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.86      0.85       200
           1       0.86      0.83      0.85       200

    accuracy                           0.85       400
   macro avg       0.85      0.85      0.85       400
weighted avg       0.85      0.85      0.85       400
```



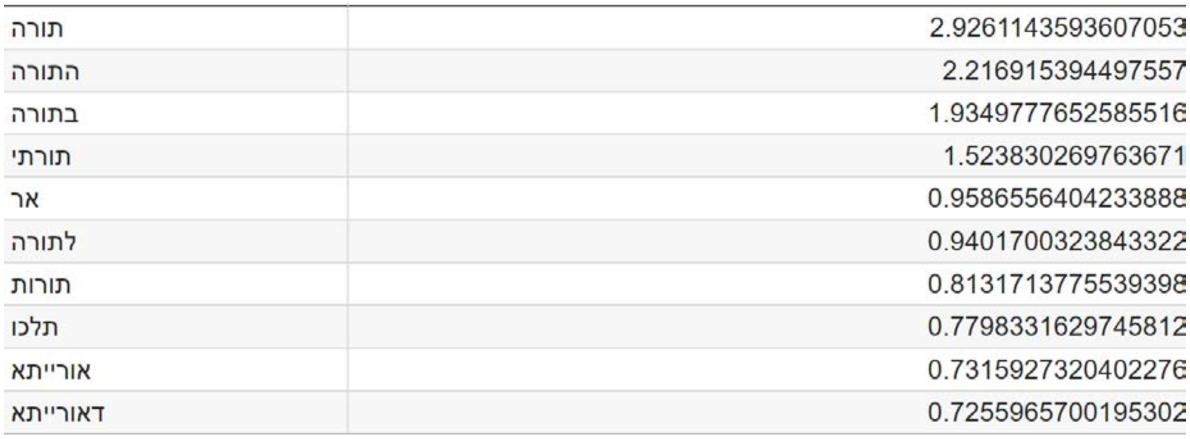
תורה	1.88557161473863
בתורה	1.7513817319858984
חכמה	1.1147548904169944
ללמוד	1.0971522021731168
הלכות	1.0235339265841212
התורה	1.0051676316436446
תורתך	0.9984722855095057
למוד	0.9293945542690933
המדרש	0.8462864728136692
בלמוד	0.8020255526913833

For the topic “למוד” (Learn), precision, recall, and f1-score were all between 83% and 86%. This was a good result as this was the first model to be created and proved to us that the project could be improved and better scores can be achieved with better machine learning models. The top words based on coefficient scores usually include similar spellings of the word or words with added prefixes. However, here the top two words are תורה and not למוד. This makes sense as when we use the word “למוד”, it typically refers to תורה learning.

תורה



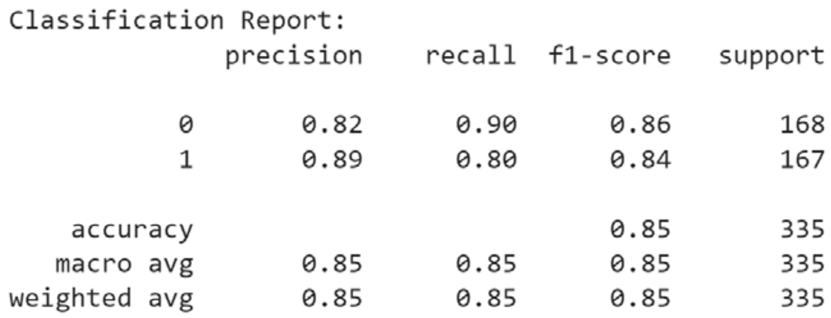
```
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.97      0.89       179
           1       0.97      0.80      0.87       179

    accuracy                           0.89       358
   macro avg       0.90      0.89      0.88       358
weighted avg       0.90      0.89      0.88       358
```



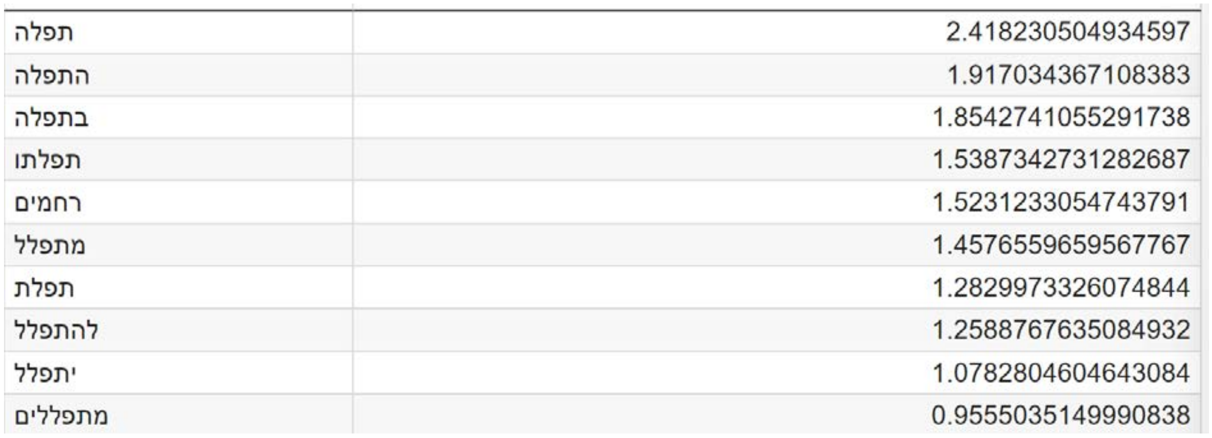
תורה	2.9261143593607053
התורה	2.216915394497557
בתורה	1.9349777652585516
תורתי	1.523830269763671
אר	0.9586556404233888
לתורה	0.9401700323843322
תורות	0.8131713775539398
תלכו	0.7798331629745812
אורייתא	0.7315927320402276
דאורייתא	0.7255965700195302

For the word “תורה” (Torah), the top words include similar spellings of the word or words with added prefixes. These words appear six times in this topic’s top ten most important words. This time the Aramaic translation of the word “תורה” appears twice. It is also interesting to note how high the values are, the first value here is almost 3, which shows how important the word is to the model. It is also interesting to note the high precision of the topic. This model was very successful in classifying texts that belong to the topic.

תפלה



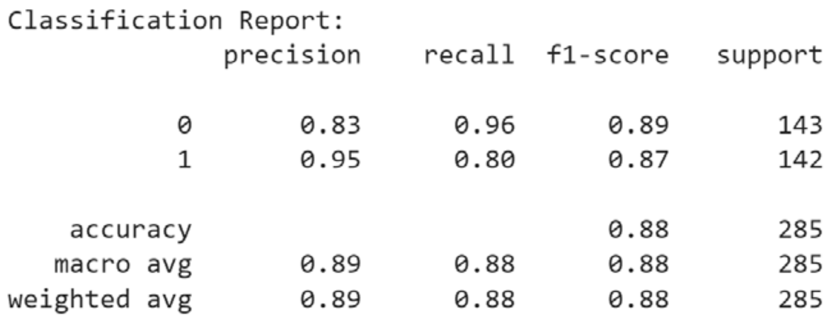
```
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.90      0.86       168
           1       0.89      0.80      0.84       167

    accuracy                           0.85       335
   macro avg       0.85      0.85      0.85       335
weighted avg       0.85      0.85      0.85       335
```



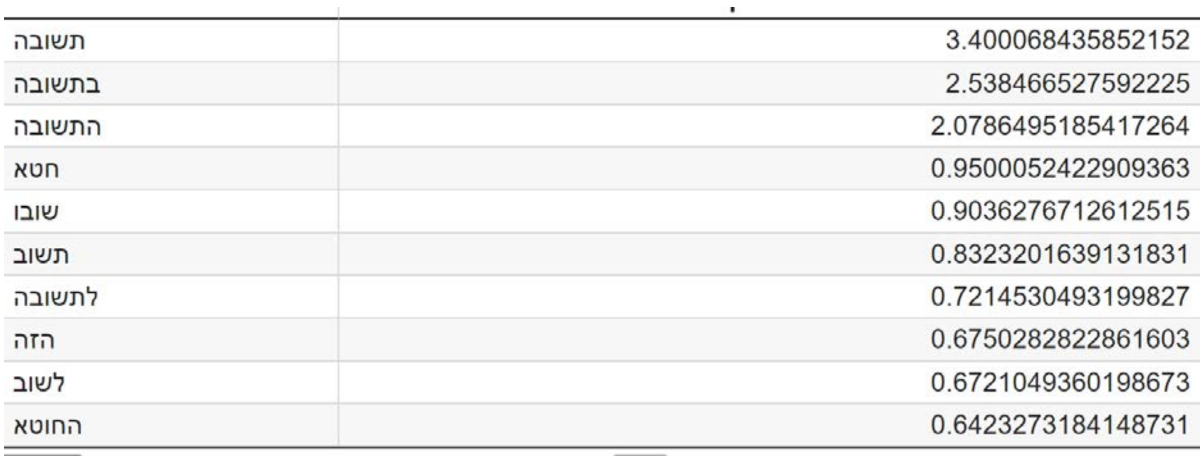
תפלה	2.418230504934597
התפלה	1.917034367108383
בתפלה	1.8542741055291738
תפלתו	1.5387342731282687
רחמים	1.5231233054743791
מתפלל	1.4576559659567767
תפלת	1.2829973326074844
להתפלל	1.2588767635084932
יתפלל	1.0782804604643084
מתפללים	0.9555035149990838

For the word “תפלה” (prayer), the top words include similar spellings of the word or words with added prefixes. This happened 20 times in the top 25 words! The word “mercy” appears once, and “please” appears once in the top 25.

תשובה



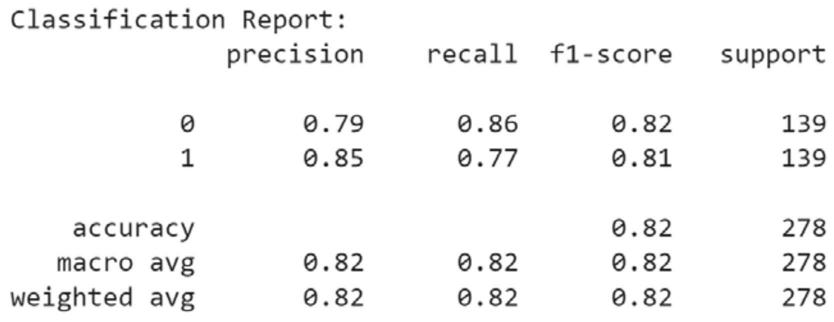
```
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.96      0.89       143
           1       0.95      0.80      0.87       142

    accuracy                           0.88       285
   macro avg       0.89      0.88      0.88       285
weighted avg       0.89      0.88      0.88       285
```



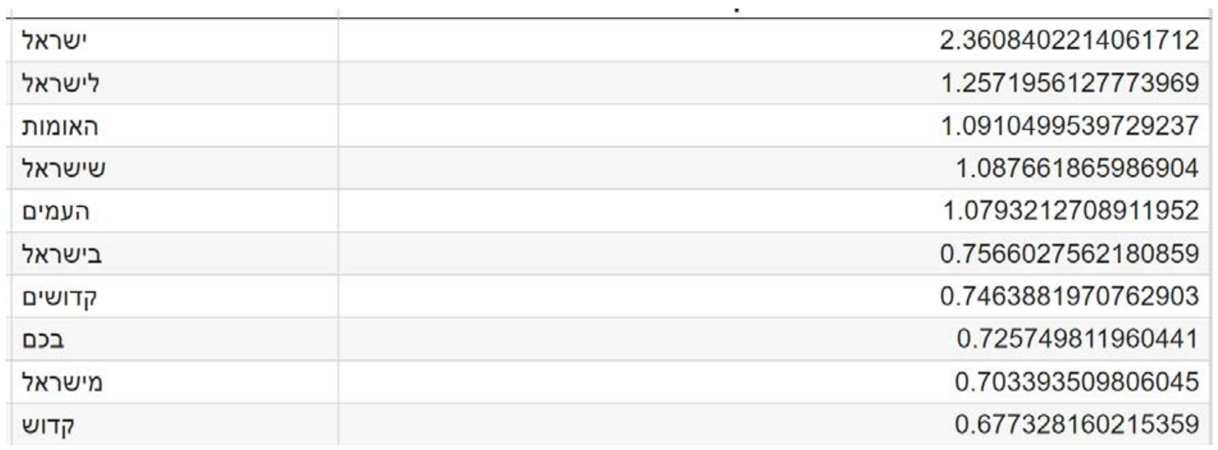
תשובה	3.400068435852152
בתשובה	2.538466527592225
התשובה	2.0786495185417264
חטא	0.9500052422909363
שובו	0.9036276712612515
תשוב	0.8323201639131831
לתשובה	0.7214530493199827
הזה	0.6750282822861603
לשוב	0.6721049360198673
החוטא	0.6423273184148731

For the word “תשובה” (repentance), the top words include similar spellings of the word or words with added prefixes. Also, the word “sin” appears twice. The word תשובה has a score of 3.4, which is the highest single word in any of the models.

ישראל



```
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.86      0.82       139
           1       0.85      0.77      0.81       139

    accuracy                           0.82       278
   macro avg       0.82      0.82      0.82       278
weighted avg       0.82      0.82      0.82       278
```



ישראל	2.3608402214061712
לישראל	1.2571956127773969
האומות	1.0910499539729237
שישראל	1.087661865986904
העמים	1.0793212708911952
בישראל	0.7566027562180859
קדושים	0.7463881970762903
בכם	0.725749811960441
מישראל	0.703393509806045
קדוש	0.677328160215359

Looking at the words with the highest coefficients: the word ישראל, which is the topic word, and variations of the word appear five times out of the top 25 words. This model also had a lower recall for texts in the topic, which is the worst performing model compared to the other four models.

All five models show that words with the highest coefficients are words with the same or different spellings, and they might also have prefixes or suffixes or words that relate to the topic word, like האומות (nation) with ישראל or חכמה (knowledge) with למוד. These results show that the models are reasonably accurate when classifying the topics. Therefore, these models became the benchmark for the rest of the project, and all future and more advanced models we made for the project were compared to these results.

Our final project uses more advanced transformer models. Transformers, also known as attention-based Large Language Models (LLMs), are state-of-the-art models in natural language processing. They can understand semantics and context in text. They are the underpinning technology for LLMs like ChatGPT, BingChat, and Bard. Our project uses pre-trained Hebrew versions of BERT (Bidirectional Encoder Representations from Transformers) models, which we then fine-tuned for our project. These models all performed better than the logistic regression models, with the best models scoring 0.95 f1-score. Here are the classification scores of the transformer models for one of the topics. alephBERT, BEREL, and heBERT are all different transformer models, and they all performed better than the logistic regression model.

תשובה model	**Accuracy**	**Precision 0**	**Recall 0**	**F1-score 0**	**Precision 1**	**Recall 1**	**F1-score 1**
alephBERT	0.9	0.88	0.93	0.9	0.92	0.86	0.89
BEREL	0.95	0.98	0.94	0.96	0.92	0.98	0.95
heBERT	0.92	0.94	0.92	0.93	0.91	0.92	0.92
Logistic Regression	0.89	0.97	0.85	0.9	0.79	0.95	0.87

Conclusion

Logistic Regression is a machine-learning technique for classification tasks. The goal is to find the probability of an item being part of a class. Its simplicity and ability to quickly understand the coefficients or weights make it a great baseline model for many classification projects. Its simplicity for text classification made it a prime choice as the baseline model of our Sefaria text classification project. The Sefaria text classification project can greatly help Sefaria and many Sefaria users.

For more information and a demo of our project, please visit our GitHub page: https://torahtexttopictagger.github.io/T4Project/

Works Cited

"About Sefaria." *About Sefaria*, www.sefaria.org/about. Accessed 7 May 2023.

D'Souza, Jocelyn. "An Introduction to Bag-of-Words in NLP." *Medium*, 4 Apr. 2018, medium.com/greyatom/an-introduction-to-bag-of-words-in-nlp-ac967d43b428.

Grover, Khushnuma. "Advantages and Disadvantages of Logistic Regression." *OpenGenus IQ: Computing Expertise & Legacy*, 23 June 2020, iq.opengenus.org/advantages-and-disadvantages-of-logistic-regression/.

Gupta, Mohit. "ML: Linear Regression." *GeeksforGeeks*, 17 Feb. 2023, www.geeksforgeeks.org/ml-linear-regression/.

Kanstrén, Teemu. "A Look at Precision, Recall, and F1-Score." *Medium*, 11 Sept. 2020, towardsdatascience.com/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec.

Kisselev, Dina. "A Simple Interpretation of Logistic Regression Coefficients." *Medium*, 10 Sept. 2021, towardsdatascience.com/a-simple-interpretation-of-logistic-regression-coefficients-e3a40a62e8cf.

Müller, Andreas, and Gaël Varoquaux. "Classifier Comparison." Edited by Jaques Grobler, *Scikit*, scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html#sphx-glr-auto-examples-classification-plot-classifier-comparison-py. Accessed 7 May 2023.

Pant, Ayush. "Introduction to Logistic Regression." *Medium*, 22 Jan. 2019, towardsdatascience.com/introduction-to-logistic-regression-66248243c148.

“Sklearn.Feature_extraction.Text.CountVectorizer.” *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html. Accessed 7 May 2023.

“Sklearn.Feature_extraction.Text.TfidfVectorizer.” *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html#sklearn-feature-extraction-text-tfidfvectorizer. Accessed 7 May 2023.

“Sklearn.Linear_model.Logisticregression.” *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression.fit. Accessed 7 May 2023.