“Supervised Natural Language Processing Models for Technical Support”

By Jorge LeonFrausto

TABLE OF CONTENTS

I. Introduction

Business Background

This project focuses on IT support operations at work. For technical issues, employees can present a request for assistance via various options: Phone, Chat or Website. Via phone, they can talk to a person directly for support. Chat allows them to use an internal tool which operates as a chatbot. If needed, they will be directed to an employee who will take over and assist the user via chat. The website option allows users to submit their issue and create tickets. No direct interaction takes place at submission. Most of these tickets will be handled via phone with the help desk calling the user at a specified time; others can be resolved by sending users instructions via emails.

Of interest is how we can improve the chatbot `Function`ality to better identify the product ("Application"). with which they're having any issues. In the initial analysis of the gathered data, there were 622 identified services. Therefore, a rule-based, or keyword search program can be computationally expensive to run. Additionally, all 622 checks will only locate one word. This ignored the additional words that may describe the Affected Products without referring to them by name. Words such as 'email', may seem simple and yet could point to: E-mail on the web; e-mail application; e-mail on the phone; and possibly, not at all to the service, rather to a single e-mail message.

Goals of Project

The purpose of this project is to identify a less complex model that can help steer users in the correct direction. From a business perspective, machine learning text classification algorithms are valuable when they structure and analyze text in a cost-effective manner, thereby expediting business processes and decision-making processes. (NLP R Primer). Text classification is important for customer service, which can involve routing customer requests based on the human language of the text. (NLP R Primer). Actionable output, such as redirecting, is the end goal of the project beyond the analysis of the machine learning models trained and analyzed here.

Functionally, how this looks for the user is that they will enter their description. The chat will return suggestions on what could apply to the user. As the program operates currently, the chatbot will provide three input-response exchanges before connecting users to a support employee.

II. Exploratory Data Analysis

Data was obtained from internal data corresponding to IT tickets. There were two primary sources: 1) Tickets created by users and submitted via the company IT portal; 2) Tickets created by the IT help desk for users who contacted the helpdesk directly. All data was compiled into a single dataset. Post-cleanup and preparation, the data set consisted of 28,173 observations.

a. Data Wrangling

All data in the data set had these features: `Case Title`; `Description`; `Source`. For tickets that were created by the help desk, additionally they had the '`Resolution`' feature. All fields were combined into one single feature.

b. Data Cleaning

Data Corrections

By manually cleaning the data row-by-row basis – further explored in section II -, I corrected some of the inconsistencies and errors in the data. Given the nature of the data, these were the most common corrections I needed to fix:

- Users selected the wrong service. e.g. We distinguish between PC – the physical computer – and Windows – the operating system. Users generally did not separate the two.
- Users selected the wrong form when filling their own tickets.
- Tickets created by advisors could have multiple issues described; however, only one ticket created.
- Non-descriptive information filled across the data set. These were removed.
- Names of services appeared in multiple ways. E.g. "Microsoft Word." "Word" and "MS Word."

Removing Non-Descriptive Data

The self-reporting system requires users to submit tickets with a short description of the issue. However, there is no check. If there is one or more characters, the submission will be accepted. In this dataset, there were many instances where the description was either a single period (.), or a variation of "na" I removed these instances where the ticket title did not provide enough information by itself. For example, "Issue with Application" did not properly describe an issue. Where a ticket titled "Teams Headset" is considered enough as we can determine user is requesting a headset to be used with the application Microsoft Teams.

Because the system operates under templates, there are standard phrases that can be attached to all descriptions. These phrases - e.g. "Issue as stated by the user"- were removed.

Non-English Information

All observations that were not in English were removed.

Target Variables

A critical element to the exploration of this process is the 'Application' feature. For those observations where this was missing, the information was manually entered.

In addition to identifying the Application, two additional target variables were calculated: Business `Function`; and, `Global Category`. Affected Features fall into only one Business `Function`s. In turn, all Business `Function`s are only part of one `Global Category`.

The next step in data cleaning was to work with free-form text variable, Description. The data was grabbed from global operations. As a result, the data included various languages. Observations that were in languages other than English were removed.

c. Data Exploration

After cleaning, and identifying the target variables. I reviewed the data set as processed.

Nineteen `Function`s were identified: Customer Care; Digital Products; User-Facing Software; Enterprise Technology Services; Global Finance; Global Human Resources; Global Quality; Global Telecommunications; Hardware Services; Information Security; Manufacturing; Microsoft Products; PC Services; Phone Services; Printer Services; Product Development; In-Car Technology; and, Supply Chain. These `Function` categories are derived from official classification under the business.

There were seven `Global Categories` identified: Software; Product Life Cycle; Hardware; Security; Business; Innovation; and, Customer Care. These categories were selected as they correspond to a business flow. Software refers to employee-used software for day-to-day operations. Life Cycle corresponds to the product life cycle from development to production. Hardware refers to servicing of employee-issued devices. Security corresponds to any requests for access rights. 'Business' referrers to general finance; and, legal and human resources `Function`s. Innovation refers to digital products – separate from physical products. And, customer care remains its sole category as this is consumer-related after sales support.

Global Category	Count	Function	Count
Software	7637	User-Facing Software	2906
		Global Telecom	1796
		Enterprise Technology	1691
		Microsoft Products	1244
Life Cycle	6527	Product Development	3383
		Manufacturing	1908
		Supply Chain	642
		Global Quality	594
Hardware	5536	Hardware Services (other)	2654
		PC Services	1324
		Phone Services	930
		Printer Services	628
Security	3609	Admin Access	2021
		Information Security	1588
Business	2733	Global HR	2193
		Global Finance	540
Innovation	1546	Digital Products	850
		In-Vehicle Software	696
Customer Care	585	CCA	585
Total			**28173**

Table 1. Global Category and Function observation counts.

III. Natural Language Operations

When working with the free-form text, it is necessary to apply transformations to create consistent results. These were the four transformations that I explored

- All text was transformed to lowercase. Else, words such as “Access” and “access” would be considered non-matching where comparisons occurred.
- Tokenized the text so that we treated each word individually.

- Removed all stop words from text. These are words that add no significant value to a sentence's meaning. E.g. "for", "in", "at."
- I reviewed both Stemming and Lemmatization. These techniques seek to reduce words to their root or base unit. Lemmatization is context-based operation. However, given the relatively short nature of the text, both gave me the same result. I kept stemming as the chosen technique.
- I reviewed N-Grams. N-grams combine *n* number of words in to tuples. For example, a bi-gram (or 2-gram) combines two words as one. This is generally useful in finding entity names such as "United States," or "United Nations." I explored transforming the data into bi-gram and tri-gram. However, the initial results were significantly worse than using single words. I did not use the n-gram technique in this project.

IV. Model Planning

a. Input Features

Global and Function Keywords

With the keywords identified, the calculation was made to find if any given keyword was present in the description. Each keyword was assigned its own column, or feature.

The tables that follow show how many words are present in the data set by frequency. It should be noted that the word counts do not take into consideration if a word appears multiple times in the same observation. i.e. A word with a frequency of 1,000 does not equate to the word being present in 1,000 observations.

GLOBAL KEYWORDS	COUNT
Over 1000	24
Over 500	45
Over 100	137
Over 50	251
Under 50	0

Table 2: Global Keyword Counts

FUNCTION KEYWORDS	COUNT
Over 1000	8
Over 500	27
Over 100	113
Over 50	51
Under 50	8

Table 3: Function Keyword Counts

Combination Keywords

An additional set of words were selected as possible input for the models. This was a combination of the Global Keywords and the Function Keywords. 190 keywords were selected. I decided to use 50% of the words from the Global Keywords and 50% from the Function Keywords.

Some keywords appeared in both Global Keywords and Function Keywords. When this occurred, I kept the word in the Function Keyword and removed it from the Global Keywords. Where a keyword appeared under more than one Function, the keyword was removed from duplicates.

Components

In addition to the keywords, there were ‘component’ sets that were created by combining either 3- or 5- keywords and reporting their total. For example, the first component column for the three-keyword model reported the total of most frequent words 1 – 3; the second component sum of keywords 4 to 6; and so forth. The same calculation was made for 5-keyword components. The calculations were only performed on the Global Keywords.

The graphic below shows how nine features would be combined into three using the 3-Keyword Component calculations.

Original Data:

ID	KW-1	KW-2	KW-3	KW-4	KW-5	KW-6	KW-7	KW-8	KW-9
1001	1	1	1	1	1	1	0	0	0
1002	0	1	0	1	0	1	1	1	1
1003	1	0	1	0	0	1	0	1	1

Component Data:

ID	C-1	C-2	C-3
1001	3	3	0
1002	1	2	3
1003	2	1	2

Figure 4 : Hypothetical 3-keyword component transformation from Global Keywords.

The machine learning models were fitted using either the Global Keywords; `Function` Keywords; Combination Keywords; 3-Keyword Component ("3KC"); or, the 5-Keyword Component ("5KC") data. No models were fitted using data from more than one of these calculated sets.

b. Word Banks

Each MLM type was created using subsets of the total word banks available. 207 for `Function` Keywords; 152 for 3KC, representing 254 keywords; 91 for 3KC representing 455 keywords. Knime models used 190 of the 457 for Global Keywords. TensorFlow models were built using up to all 457 keywords.

The models were created using subsets of the word banks to test how various sets of keywords performed. For example, the Global Keyword consists of 457 keywords. However, looking at the frequencies I used only the most frequent words with at least 1000 observations; for the next build those with 900 or more, etc. For Function Keywords, I fed the data using the most frequent words. Then the second and most common words. The word bank was increased by about 19 on each run – aligning with the 19 identified `Function`s.

The table below shows the bank word sizes. I used a different word count splits for the Random Forests.

Global Keywords (Random Forests)	Global Keywords	`Function` Keywords	Combination Keywords	3-Keyword Component	5-Keyword Component
52	22	20	38	30	50
75	33	39	76	75	75
99	48	58	114	93	95
125	56	77	152	135	135
225	64	96	190	150	170
300	77	115		219	220
457	94	134		324	270
	103	153		384	325
	121	169		456	385
	136	184			420
	160	195			455
	190	207			

Table 5: Word Bank breakdown by input source type.

c. Machine Learning Model Algorithms

I tested data from those calculated sets and modeled five types of models. Decision Trees; K-Nearest Neighbor; Random Forest; and Boosted Gradient Tress were created using Knime. The fourth type was a Random Forest using Keras library in TensorFlow.

Random Forests were trained and developed in a Jupyter notebooks and Knime. All other machine learning models were created in Knime using a no-code interface.

d. Target Variables

All models were trained using both the `Function` and the `Global Category` as target variables.

e. Trained Models Count

The graph below shows how the various elements are combined to create the models trained. A total of 592 models were trained for this project.

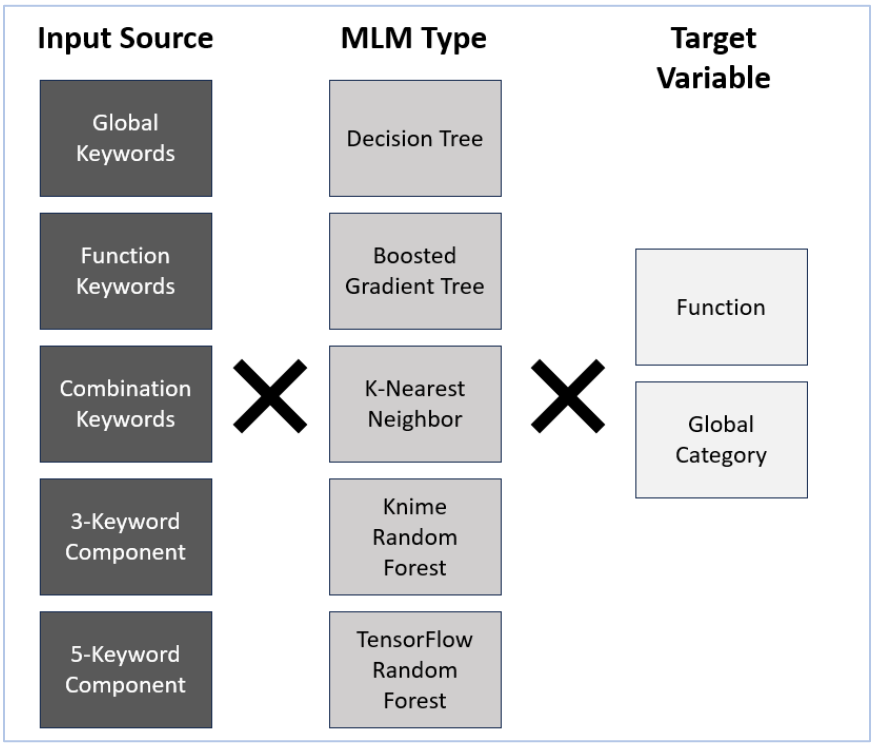

Figure 6: Diagram Showing Input-Learning Model-Target Combinations

V. Machine Learning Models – Variable Impact

The next step following training all models, I proceeded to review how the models' performance was affected by the various factors.

a. Input Source

The tables below show the models' performance when broken down by Input and machine learning algorithm.

	Decision Trees	Gradient Boosted Tress	K-Nearest Neighbor	Knime Random Forest	TensorFlow Random Forests	Accuracy
3-Keyword Component	0.5953	0.6638	0.6200			0.6274
5-Keyword Component	0.5689	0.6406	0.6031			0.6042
Combination Keywords	0.6268	0.6622	0.6256	0.5783	0.5958	0.5973
Global Keywords	0.5970	0.6383	0.6046	0.5927	0.6184	0.6079
`Function` **Keywords**	0.6323	0.6606	0.6281			0.6403
Summary	0.6027	0.6518	0.6156	0.5867	0.6089	0.6110

Table 7: Cumulative Model Performance for `Function` as target variable.

	Decision Trees	Gradient Boosted Tress	K-Nearest Neighbor	Knime Random Forest	TensorFlow Random Forests	Accuracy
3-Keyword Component	0.6787	0.7189	0.6965			0.6274
5-Keyword Component	0.6596	0.7098	0.6898			0.6042
Combination Keywords	0.7145	0.7245	0.7134	0.5783	0.5958	0.5973
Global Keywords	0.6946	0.7079	0.6933	0.6911	0.7103	0.6079
`Function` **Keywords**	0.7165	0.7255	0.7190			0.6864
Summary	0.6912	0.7166	0.7016	0.6477	0.6626	0.6795

Table 7: Cumulative Model Performance for Global Category as target variable.

Looking at the performance of the inputs, the worst performing input source is the 5-Keyword Component. Interesting to the analysis is that the Global Keywords were the next lowest performers.

The reason why the 5-Keyword component underperforms is because its an aggregated variable. As such, there is a loss of what specific keywords contribute to the calculation. The main reason why Global Keywords appear to underperform is that the most common words apply, as the name imply, globally across the `Function`s.

b. Word Bank

The tables below show how models performed by word bank (size) and Model Type. The number of words were grouped in sets of 50 for these tables.

Word Bank	Decision Trees	Gradient Boosted Tress	K-Nearest Neighbor	Knime Random Forest	TensorFlow Random Forests	Accuracy
< 50	0.5702	0.6020	0.5782	.05668	0.5753	0.5777
51 - 100	0.6316	0.6815	0.6349	0.5928	0.6111	0.6209
101 - 150	0.6616	0.7107	0.6604	0.6073	0.6298	0.6429
151 - 200	0.6508*	0.7099*	0.6618	0.6148	0.6239	0.6467
201 - 250	0.6054*	0.7016*	0.6488			0.6612
251 - 300	0.6232	0.7171	0.6636*	0.6164	0.6374	0.6406
301 - 350	0.6298	0.7275	0.6697			0.6757
351 - 400	0.6119	0.7080	0.6589			0.6596
400+	0.6351	0.7341	0.6739	0.6112	0.6678	0.6533
Summary	0.6027	0.6518	0.6156	0.5867	0.6089	0.6110

Table 8: Cumulative Performance – `Function` as target variable. Effect of Word Bank.

Word Bank	Decision Trees	Gradient Boosted Tress	K-Nearest Neighbor	Knime Random Forest	TensorFlow Random Forests	Accuracy
< 50	0.6648	0.6817	0.6708	0.6367	0.6217	0.6550
51 - 100	0.7193	0.7354	0.7237	0.6545	0.6745	0.6891
101 - 150	0.7414	0.7624	0.7443	0.6378	0.6609	0.6948
151 - 200	0.7278	0.7640	0.7441	0.6958	0.7143	0.7252
201 - 250	0.7023	0.7518	0.7324			0.7288
251 - 300	0.7070	0.7676	0.7378	0.7014	0.7292	0.7227
301 - 350	0.7109	0.7735	0.7456			0.7433
351 - 400	0.6965	0.7716	0.7366			0.7349
400+	0.7050	0.7801	0.7404	0.7313	0.7532	0.7421
Summary	0.6912	0.7166	0.7016	0.6477	0.6626	0.6795

Table 9: Cumulative Performance – Global Category as target variable. Effect of Word Bank.

These tables help in identifying the word ranges where overfitting occurs based on the word bank used for models. Excluding the Random Forests models, most overfitting for `Function` occurs somewhere between 151 and 250 words. For `Global Categories`, the overfitting occurs at a higher number of words, around 200 or more words.

When the model type is removed from the analysis above, the following scatter plots are created. These scatter plots exclude the Random Forest models. Visually, there are a few trend lines that appear with a steep increase in slope and a flattening around the 150- to 200-word mark. The wide spread that can be seen for words banks over 250 can be attributed to the 3-Keyword and 5-Keyword Component effects of aggregation.

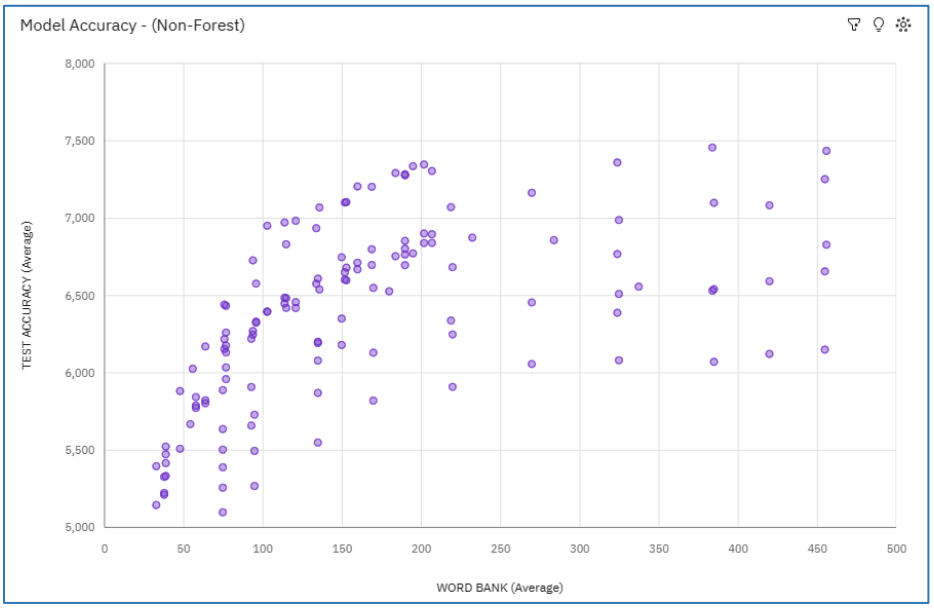

Figure 10: Model Performance by Word Bank – `Function` as target variable.

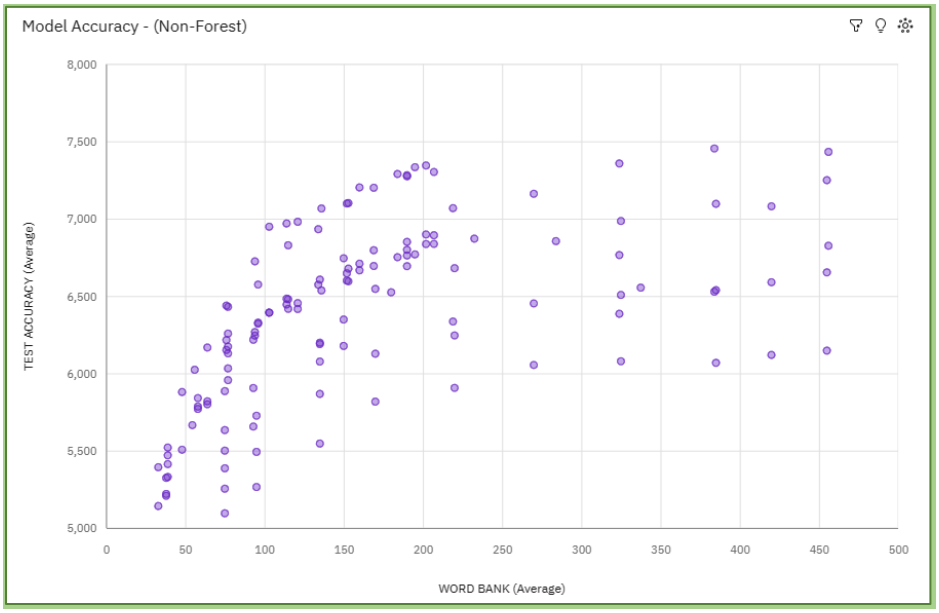

Figure 10: Model Performance by Word Bank – Global Category as target variable.

c. <u>Random Forests: Depth and Trees</u>

Word Bank	Depth: 20		Depth: 25		Depth: 50		Summary
	Trees: 25	Trees: 50	Trees: 25	Trees: 50	Trees: 25	Trees: 50	
38	0.4996	0.4973	**0.**5059	0.5059	0.5098	0.5111	0.5048

52	0.5379	0.5367	0.5541	0.5552	0.5640	0.5670	0.5525
75	0.5448	0.5428	0.5602	0.5633	0.5930	0.5976	0.5670
76	0.5648	0.5679	0.5858	0.5884	0.6131	0.6172	0.5895
99	0.5630	0.5656	0.5872	0.5892	0.6288	0.6301	0.5940
114	0.5670	0.5715	0.5918	0.5933	0.6355	0.6361	0.5992
125	0.5808	0.5868	0.6094	0.6072	0.6494	0.6433	0.6128
150	0.5877	0.5939	0.6016	0.6064	0.6653	0.6678	0.6204
152	0.5723	0.5775	0.6015	0.6058	0.6581	0.6618	0.6128
190	0.5947	0.5940	0.6236	0.6180	0.6728	0.6705	0.6289
225	0.5811	0.5890	0.6054	0.6062	0.6665	0.6680	0.6193
300	0.5929	0.5822	0.6129	0.6175	0.6735	0.6827	0.6269
457	0.5986	0.5964	0.6338	0.6330	0.6856	0.6898	0.6395
Summary	0.5682	0.5694	0.5910	0.5920	0.6322	0.6342	0.5978

Table 11: Random Forest performance. `Function` as target variable.

When the target variable is `Function`, increasing the number of trees does not appear to cause overfitting on the models. Yet, the increase in accuracy is very minimal, indicative that 25 trees is the better choice. Here, increases the depth to 50 has a significant increase over models with a depth of 25. On average, where the target variable is `Function` a depth of 50 appears to be the better selection.

Word Bank	Depth: 20		Depth: 25		Depth: 50		Summary
	Trees: 25	Trees: 50	Trees: 25	Trees: 50	Trees: 25	Trees: 50	
38	0.4996	0.4973	0.5059	0.5055	0.5098	0.5111	0.5048
52	0.6356	0.6402	0.6508	0.6500	0.6651	0.6673	0.6515
75	0.6475	0.6561	0.6622	0.6735	0.6954	0.6944	0.6715
76	0.5648	0.5679	0.5858	0.5884	0.6131	0.6172	0.5895
99	0.6664	0.6731	0.6940	0.6910	0.7230	0.7203	0.6946
114	0.5670	0.5715	0.5918	0.5933	0.6355	0.6361	0.5992
125	0.6864	0.6783	0.6921	0.6885	0.7249	0.7275	0.6996
150	0.6724	0.6723	0.6972	0.7011	0.7430	0.7521	0.7063
152	0.5723	0.5775	0.6015	0.6058	0.6581	0.6618	0.6128
190	0.5947	0.5940	0.6236	0.6180	0.6728	0.6705	0.6289
225	0.6721	0.6691	0.6872	0.6815	0.7577	0.7628	0.7050
300	0.6613	0.6876	0.7012	0.7060	0.7675	0.7683	0.7153
457	0.7244	0.7148	0.7337	0.7315	0.7713	0.7781	0.7423
Summary	0.6272	0.6297	0.6476	0.6478	0.6871	0.6896	0.6548

TABLE 12: Random Forest performance. Global Category as target variable

For `Global Category`, the findings differ from those in the `Function` analysis. The tables reveal that the keeping the number of trees at 25 is a better choice than 50. At the lowest and highest word counts, the number of trees at 50 underperforms the 25-tree models; this indicates overfitting. An increase in depth does not show any overfitting indicators. However, while the overall performance increases, so does the computational cost of the models. So, upon initial review, it may not be worth increase the depth past 25 levels for an optimal model selection.

d. Machine Learning Model

Where all other factors are eliminated, the spreads of the five machine learning model algorithms selected for this project are in the graphs below.

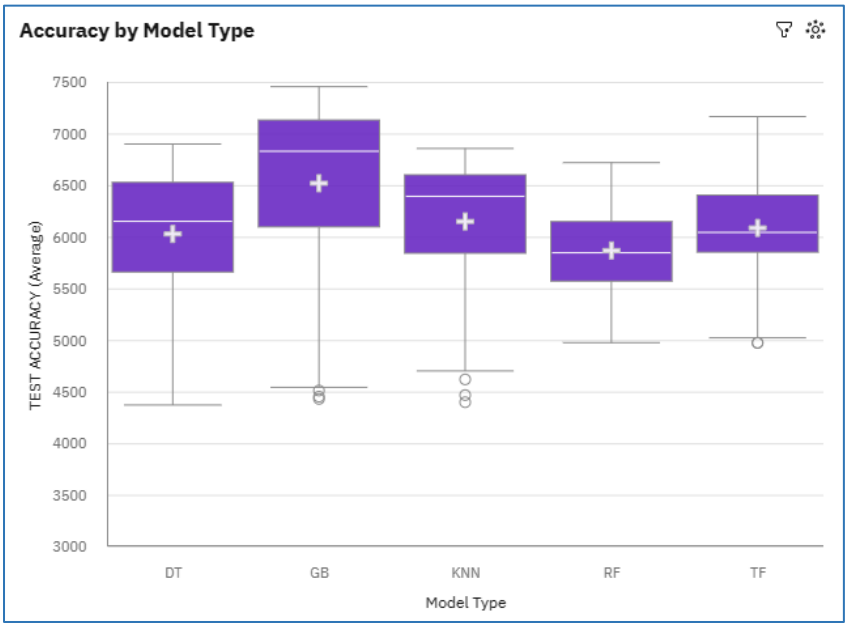

Figure 13: Performance of Machine Learning Models – `Function` as target variable.

For training when the target variable is `Function`, Gradient Boosted Trees performed the best. The worst, by the Inter-Quartile Range are the Knime Random Forests. However, the worst model is a Decision Tree Model.

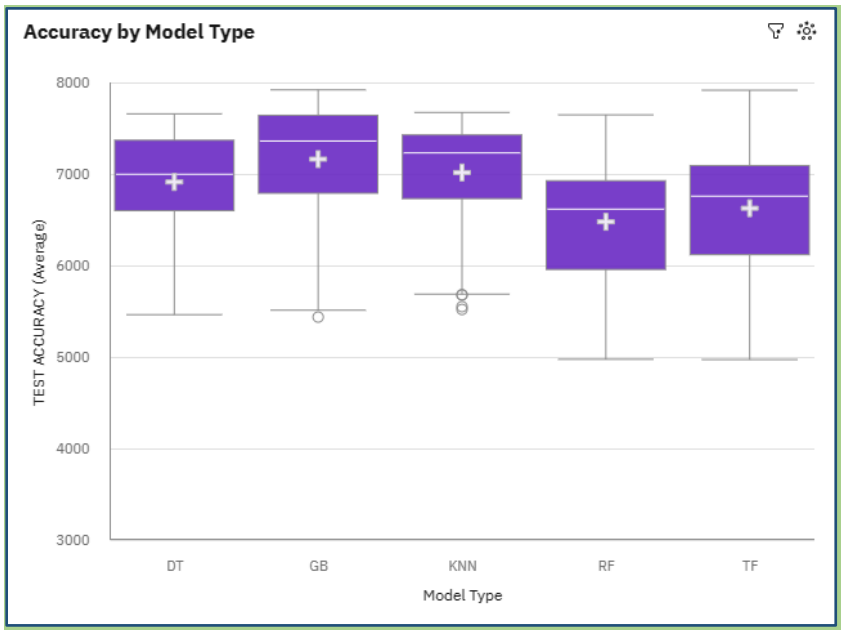

Figure 14: Performance of Machine Learning Models – `Function` as target variable.

When the target variable is `Global Category`, both the best and worst models are for TensorFlow Random Forests. These reflect the wide spread of the results for those models. The most compact model, as with `Function`, are the K-Nearest Neighbors. And, like above, the Gradient Boosted Trees are the best models overall where `Global Category` is the target variable. Here, Decision Trees perform relatively close to the Gradient Boosted Tree models.

VI. In-Depth Model Performance Analysis

Evaluation was executed on all models after running them to determine which could be deemed acceptable for analysis. The following analysis was applied to target `Function` and `Global Category` models.

a. Model Variants

A model is considered a variant when the value of one of its construction variables change. Only one variable can change for this to be considered a variant. i.e. The word bank size; or, for Random Forests, either the depth or the number of trees can change among variants. E.g. A Gradient Boosted Tree with word bank of 150 Global Keywords is considered a variant of a Gradient Boosted Tree with 99 Global Keywords. It is not a variant of a Gradient Boosted Tree with 150 Function Keywords. Input source – which keyword set is used – is not a variable that can change for models to be considered variants.

b. Stages of Analysis

There were four stages to this analysis: 1) Minimum performance; 2) Compared variants in models by keyword; depth (for Random Forests); and, trees (for Random Forests) by looking at changes in accuracy. 3) Best performers by model type; 4) Final comparison of all "finalists" to choose the final recommendation.

1. Minimum Performance

When reviewing model performance in the early stages of training – specifically, only training Decision Trees with Global Keywords – it was determined that models with an accuracy below 0.600 generally were unable to identify all Functions. A similar occurrence existed with Global Category at around 0.5000. Therefore, the first eliminator was to not consider models below this minimum acceptable performance.

2. *Variant Comparisons*

A general formula was used when comparing models by word bank. Assuming Model 1 and Model 2 are variants; with Model 2 using the larger word bank. The following formula was used to determine if the model using the larger word bank should be eliminated:

$$\frac{\Delta_a}{\Delta_\omega} = \frac{a_2 - a_1}{\omega_2 - \omega_1} > 0.0050$$

This reads: the change in accuracy divided by the change in word count should be greater than 0.0050 – or 0.05% -. If increase is less than or equal to 0.0050, Model 2 is eliminated.

i. Depth and Trees – Random Forests

For depth, there were three values 20, 25, or 50 when training the Random Forests. The generally guidelines followed were that a model should increase accuracy by 0.01 from 20 to 25 and; 0.02 when moving from 25 to 50. When considering the number of trees I applied the 2% (0.02) threshold when increasing the number of trees from 25 to 50.

3. Best Models and Best Models by Type

The best models were grouped together for an analysis to look further into whether they should still be considered despite any of the calculations above marking them for removal. My general findings is that I could determine why these models were not selected – particularly due to model complexity.

After the initial calculations, the top 5 or 6 models by type were chosen and grouped as follows: Decision Trees and K-Nearest Neighbor; Gradient Boosted Trees; and, Random Forests.

I looked at how the models compared to each other based both on increased complexity and accuracy. One or two models for each of the three groups were chosen for a final comparison.

4. *Final Comparison*

Having chosen the best models by type, I reviewed the recommended models and a final selection was made. The same process as when broken down by type was applied, reviewing model complexity and accuracy.

VII. Recommended Models

Function Models

These were the final Function Models considered for recommendation:

Model ID	Type	Input	Word Bank	Depth	Trees	Accuracy
53	Gradient Boosted Tree	Global Keywords	103	10	24	0.6948
202	TensorFlow Random Forest	Global Keywords	150	50	25	0.6653
284	Decision Tree	Combination Keywords	114	17		0.6483
294	Gradient Boosted Tree	Combination Keywords	114	10	24	0.6969

Among the models trained for target variable Function, the recommended model is Model 294. Model 294 has these attributes:

- Gradient Boosted Tree
- Uses Combination Keywords with a bank size of 114.
- Its accuracy is 0.6969

For comparison, the best Function-targeting model was Model 66. Model 66 is also a Gradient Boosted Tree which uses 384 Global Keywords; and, has an accuracy of 0.7454.

Global Category Model

These were the final Global Category Models considered for recommendation:

Model ID	Type	Input	Word Bank	Depth	Trees	Accuracy
54	Gradient Boost Tree	Global Keywords	121	10	24	0.7547
67	Gradient Boost Tree	Function Keywords	169	10	24	0.7701
87	Knime Random Forest	Global Keywords	125	50	50	0.7208
202	TensorFlow Random Forest	Global Keywords	150	50	25	0.7583
284	Decision Tree	Combination Keywords	114			0.7374
295	Gradient Boost Tree	Combination Keywords	152	10	24	0.7642

Where the target variable is the Global Category, Model 54 is the recommended model. Model 54 has these attributes:

- Gradient Boosted Tree
- Uses Global Keywords with a bank size of 121.
- Model accuracy 0.7547

By comparison, the best performing Global Category-targeting model was Model 67. Also, a Gradient Boosted Tree, using 456 Global Keywords. And, an accuracy of 0.7919

VIII. Discussion

These models were recommended because, as a general approach, an optimal or preferred models sits at a balance between complexity and accuracy. In this project, simplicity is preferred

as speed is the goal of the models, rather than accuracy. This was reflected in the thresholds used when eliminating models.

Looking at the finalists, the models that I reviewed all had word banks under 200. The best performing models were excluded because they were too computationally costly. In the future, if there is no preference towards speed, I expect that models closer to the 200-word bank mark will be optimal.

The choice of which keywords to feed the model showed that Function Keywords by themselves are not enough. From the results, they're in a way likely to "overfit" because they're so specific. This is why I believe the Combination keyword was the one present in the chosen model for `Function` as a target variable. Additionally, the `Global Category` used the Global Keywords.

Gradient Boosted Trees proved to outperform all other algorithms used when training models. I believe that given the nature of the project, the Random Forests prove too complex in nature. As with the word bank analysis above, if speed is not prioritized over complexity, Random Forests can be used to focus on accuracy.

Going Forward

There are a few things that can be done to improve on the results of the existing project; or, expand on its applications.

Data Size and Sampling

While the data size of over 26,000 observations, the three least-common Functions had a total of 1,719 observations – or about 6.6%. An explicit effort to try and balance the distribution

of all functions should be done. The real distribution for those functions are closer to 3% for the three least-common Functions. It required me to manually search for cases that were already classified to them. Understanding the low frequency of requests, any future developments may require the elimination of the least-common Functions.

Keyword Calculations

Related to the frequency topic above, the calculation of Function Keywords is dependent on that sampling. I expect that we could very likely see changes to the list of most common words by Functions. It should be noted, I do not think top 3 or 5 by Function will likely be affected.

When computing Combination Keywords, I decided to do an even split between Global Keywords (50%) and Function Keywords (50%) in the composition of this set. Additional exploration can take place by manipulating the percentages.

Target Variable Optimization

This project focused on identifying the Application, or Function for the customer inquiry. Focusing on actionable output, project data can be used to identify whether a request can be completed by the user without need to contact the help desk.