Github README Classification - ENSF612 Final Project

MICHAEL KISSINGER, University of Calgary, makissin@ucalgary.ca, Canada
BRANDON QUAN, University of Calgary, brandon.quan@ucalgary.ca, Canada
MEET PANDYA, University of Calgary, meet.pandya@ucalgary.ca, Canada

1 SUMMARY OF WORK DISTRIBUTION

Here, we will summarize the work distribution between the three members of this team. In order to fetch 1000 new datasets (Github Readme Sections), we first needed to fetch many random Github Repositories. The three of us worked on putting together a simple python program to fetch around 300+ github repositories. Next, we each took up around 100+ github repositories to look through their Readmes and label the different sections manually as described in the original paper [1]. Overall for the labelling of the dataset we each ending up labeling a little over 335 Github Readme sections individually. Using performance metrics of accuracy, precision, recall, and F1, different combinations of models and hyperparameters were tested against the eight categories. What, Why, How categories were tested by Meet, When, Who, Other categories were tested by Brandon, and References Contribution were tested by Michael.

Section	Writer
1,2,4.1,5.1,5.6, Discussions, Conclusion, Appendix C	Meet Pandya
3,4.2,5.4,5.5,5.6, Discussions, Appendix B and D	Brandon Quan
3,5.2,5.3,5.6, Discussions, Appendix A	Michael Kissinger

Table 1. Report Work Breakdown

2 DATABRICKS NOTEBOOK LINKS

- The first notebook below has the code for training of the models, displaying different accuracy score and confusion matrices.
- https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/3634097844023146/150675221134314/8712538996775317/latest.html
- The next notebook below shows how Data Preprocessing was achieved
- https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/1222114862673243/2771296308904893/3779371744576095/latest.html
- The next notebook below shows how Misclassification data was retrieved by using PySpark
- https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/436154438772259/3379983875769477/40505063256021/latest.html

3 ABSTRACT

3.1 Context

Github is one of the most popular code repository platforms out there, and README files in a repository are a critical component that allow the author/developer of the repository to describe its contents. Even though Github README files are suppose to provide sufficient information about the contents of the repository, it is not always the case.

Authors' addresses: Michael Kissinger, University of Calgary, makissin@ucalgary.ca, Canada; Brandon Quan, University of Calgary, brandon.quan@ucalgary.ca, Canada; Meet Pandya, University of Calgary, meet.pandya@ucalgary.ca, Canada.

3.2 Objective

In this paper, we referenced the work that was done in the paper **Categorizing the Content of Github README Files**[1][3]. The purpose of this paper was to add 1000 new data points and extend the original paper's dataset. Afterwards, we would train our dataset on different algorithms to see if the resulting models improved on the metrics presented in the original paper's results.

3.3 Method

From the new data that we collected and manually labeled, the goal was to train a few classification algorithms to see which one performs the best. Next, we ran the original dataset from the original paper through our classification algorithms to see the accuracy scores of the best performing models. Finally, we combined our new dataset with the original dataset and ran it through the same algorithms to see if whether the model's performance improved or not.

3.4 Result

Upon collecting and labelling the data extracted from random Github repositories using the Github API, it was found that there was substantial agreement between the labellers and that the new dataset matched closely to that of the original dataset.

One thing we noticed was that most common section found in a Github README file was the "How" category. This makes sense because Github, as a source control platform, has millions of developers who publish their code and provide instructions on how to run it in case someone wants to leverage it.

Upon preprocessing the datasets, a set of statistical and heuristic features were generated and used to train SVC models that had an accuracy of 0.8950 on the original dataset and 0.8864 on the combined dataset. The hyperparameters of these models were with a maxIter parameter of 30 and default parameters, respectively.

There were some misclassifications in the model, mostly stemming from certain key words triggering an alternative classification, not enough words to form a proper classification, or the algorithm not being able to interpret unprocessed text that made it through the preprocessing steps.

3.5 Conclusion

Throughout the steps outlined in the paper, we were able to follow most of the work done in the original research paper and apply it to a new set of 1000 datapoints. The features extracted from the manual labelling of the new dataset were successfully used to train SVC models that performed comparably on the original dataset and combined dataset.

Via this project, we have come to a conclusion as to how critical Github README files are. Having manually labelled 1000+ new data points as a group, we saw plenty of repositories that could be improved significantly. By applying Machine Learning to Github README files, it can be a very powerful way to easily identify what is contained in a README file. The ability to easily split up different sections of a README file into the given categories can also provide end-users with convenient access to useful info, if it exists.

ACM Reference Format:
Michael Kissinger, Brandon Quan, and Meet Pandya. 2021. Github README Classification - ENSF612 Final Project. 1, 1 (December 2021), 68 pages.

4 INTRODUCTION

4.1 Motivation

The motivation behind this problem came from the fact that the three of us are all aspiring software developers and are familiar with Github and the README files present in its repositories. When going through the **Categorizing the Content of Github README Files** paper, its contents caught our attention. Classification is one of the most common problems solved by machine learning, since it is able to classify data very effectively. Knowing this, we were really curious how classification of Github README sections could be achieved via machine learning.

4.2 Background

The original paper had set out with the goal of understanding the content of Github README files and developing a classifier that can categorize the sections of a README file. Upon developing a classifier, different features and their affect on the model's performance were evaluated and used to tune the model further. Finally, the original paper polled developers to see if classifying the sections in a Github README made a noticeable difference in understanding the repository's contents.

To establish the content of README files, the paper reported on a qualitative study of 393 GitHub README files containing a total of 4,226 sections. For each of the sections, the original paper had manually annotated each section in accordance to a schema developed based on the initial analysis of the study.

The schema consisted of eight categories that included What, Why, How, When, Who, References, Contribution, and Other. Upon finishing the manual annotation of the sections, the respective frequencies of each category were evaluated and then used to develop a set of features that can be used to train a classfier to predict categories of sections in the README files.

The features generated for each Github README section included statistical and heuristic features. Using these features, the classifier's performance was evaluated on the manually-annotated dataset, and the most important features for distinguishing the different categories of sections were observed. Based on the generated features, a support-vector machine algorithm was found to be particularly effective at the task of classifying sections.

Finally, a survey was developed to evaluate the usefulness of the classification by presenting the automatically determined classes to label sections in GitHub README files to software developers. The survey found that the classification was helpful in the process of streamlining the most relevant information within a repository to new developers.

5 RESULTS

5.1 Data Collection and Data Labeling

5.1.1 Approach.

Data Collection. For data collection, the first step was to retrieve enough random Github repositories in order to have at least 1000 Github README sections to label. In the original paper, the researchers mentioned that they used the Github API to fetch the repositories[2]. We decided to do the same thing by writing up a simple Python script to fetch repositories.

We broke down the Python script in 2 tasks. The first task was to fetch random set of Github repository URLs. The next task was to compare the newly fetched URLs with the original dataset and only pick out the ones that are not found in the original dataset. In order to make the Github API call, we leveraged Python's built-in **Requests** library. And, in order to parse the original paper's csv file with the dataset, we used Python's **Pandas** library. The Python code for the first step to fetch Github repositories can be seen below.

```python
def callgithub():
    URL = "https://api.github.com/repositories?since=100495"
    r = requests.get(URL)
    data = r.json()
    for i in data:
        print(i['url'])
    return data
```

Listing 1. Fetch Github Repos

The above code fetches an array of JSON objects containing details about each random repository fetched. In the URL variable, the **since** parameter informs the API to only fetch repositories with ID greater than the one passed in. We then parse the JSON object and loop through the list of JSON to print out just the repository URL. Finally, the data variable is returned.

The next step is to compare the repository URLs from **callgithub()** function and compare with the original paper's dataset which is available in the form of a csv file. The Python code for the second step can be seen below.

```python
def readCSV():
    df = pd.read_csv('../READMEClassifier/input/dataset_1.csv')
    githubData = callgithub()
    listOfRepo = []
    for i in githubData:
        for c in df['url']:
            if i['url'] == c:
                break
            else:
                listOfRepo.append(i['url'])
                break
    print(len(githubData))
    print(len(listOfRepo))
    return listOfRepo
```

Listing 2. Compare URLs with CSV

The above code reads the original dataset's csv file into a Pandas Dataframe. Next, it calls the **callgithub()** function to fetch the array of JSON objects containing newly fetched repos. Then, using a nested for loop, the function compares to see if the URL in new dataset exists in the original dataset. If it does not exist, it it then appended to a new variable (**listOfRepo**). Finally, the listOfRepo variable is returned from this function.

Data Labelling. For data labelling, we essentially split up the list of repositories we collected by approximately 1/3rd between the three members of this team. Before moving onto labelling, we created a Google Sheet to keep track of all the manual labels that we would be assigning. Once we each took up a chunk, we started looking at each repository manually. The first thing we checked was to make sure the README file was at least 2 kb in size. This was also something that was done in the original paper. The reason for this is that if a README file size is too small, there is a strong possibility that its content is not of a good quality. The next thing we looked out for was to make sure that the entire README file was in English.

Once we found useful README files, for each repository we would pick out all the different section headings that we could easily identify. We would then identify what type of repository it was. We would store these headings into their own columns in our Google Sheet, for each repo. Next, for each section heading we assigned single or multiple numerical codes.

We used the same coding schema that the original paper came up with. The categories for their coding schemas are What(1), Why(2), How(3), When(4), Who(5), References(6), Contribution(7) and Others(8). The original paper also classified the repositories as one of the following, an end-user app, a framework, a library, learning resource, or a project related to UI. We also added a column to classify the repository based on our judgement. Figure 1 below shows what the classification looks like in our spreadsheet.

B	C	D	E
URL	Repository Category	Heading	Classification
https://github.com/merb/merb	Frame	# Merb	1, 4, 6
		## Modules	3
https://github.com/rubinius/rubinius	Lib	# The Rubinius Language Platform	1
		## Code of Conduct	5
		## Issues & Support	6
		## Contributing	7
		## License	5
		## Installing Rubinius	3
		## Philosophy & Architecture	3

Fig. 1. Manual Labelling Example

5.1.2 Results. Once all the manual labelling was completed we ended up with 1226 heading and content sections. For the 1226 headings and sections, we ended up using 143 Github Repositories. The reason why we had 1226 dataset points but only 1018 distinct sections is due to the fact some sections had multiple categories assigned to them. In the table below we can see the breakdown of the dataset points by different categories labelled as their codes.

Classification Category	Number of Codes	Percentage
Code 1 Flag (What)	174	14.19%
Code 2 Flag (Why)	46	3.75%
Code 3 Flag (How)	612	49.92%
Code 4 Flag (When)	46	3.75%
Code 5 Flag (Who)	140	11.42%
Code 6 Flag (References)	177	14.44%
Code 7 Flag (Contribution)	20	1.63%
Code 8 Flag (Other)	11	0.90%
Total	**1226**	**100**

Fig. 2. Labelling Metrics for each Category

Next, we are going to discuss the quality of data labelling by talking about the percentage agreement. In order to improve data labelling quality, we each worked on one of the other labelers' list of section headings. On their list of dataset we added our labels of what we think the category a section heading belongs to. Once that was done, we manually calculated the percentage agreement and disagreement. The results for which can be seen in the table below. Between all the users there was substantial amount of agreement. Looking at the metrics, the lowest disagreement was at 5.24% while the highest disagreement was at 16.46%.

	Total Disagreements	Total Headings	Percentage Disagreement	Percentage Agreement
Meet	55	334	16.46706587	83.53293413
Brandon	19	362	5.248618785	94.75138122
Michael	36	322	11.18012422	88.81987578
Total	110	1018	10.80550098	89.19449902

Fig. 3. Percentage Agreement and Disagreement

The next approach we used to look at the quality of data labelling was to calculate the **Cohen Kappa** score. Cohen Kappa requires a more complex set of calculations, which were done manually. Once the two key variables for Cohen Kappa score were calculated, we applied that to the final equation to get the Agreement score. The Cohen Kappa score shows substantial to near perfect agreement between all the labelers.

Labelers	Cohen Kappa Score	Implication
Brandon-Michael	0.94	Near Perfect Agreement
Michael-Meet	0.91	Near Perfect Agreement
Meet-Brandon	0.71	Substantial Agreement

Table 2. Cohen Kappa Scores

5.2 How does new data compare with Original?

5.2.1 Approach. The primary metric that is used in the original study when looking at their data is the distribution of README category types, based off of the manual labeling of these categories. We followed the same approach by examining the category distribution in the original data set, the new data set, as well as an aggregate of both data sets.

5.2.2 Results. For both the original and the new data set, the distribution was very unbalanced. The largest category was "How", under which 51.08 percent of the README sections fell for the original data set, and 49.92 for the new data set. On the other end of the spectrum the least populated categories were "Contribution" and "Other" with 2.53, and 1.2 percent for the original data set, and 1.63 and 0.9 percent for the new data set.

Above we can see the percent distribution for each category. There is a very close match between almost every category for the new vs the original dataset, with all but one category being

Classification	Number of Codes	Code 1 Flag	Code 2 Flag	Code 3 Flag	Code 4 Flag	Code 5 Flag	Code 6 Flag	Code 7 Flag	Code 8 Flag	TOTAL
Original Dataset	Headings	707	116	2467	180	322	858	122	58	4830
	%	14.64%	2.40%	51.08%	3.73%	6.67%	17.76%	2.53%	1.20%	100
New Dataset	Headings	174	46	612	46	140	177	20	11	1226
	%	14.19%	3.75%	49.92%	3.75%	11.42%	14.44%	1.63%	0.90%	100
Combined Dataset	Headings	881	162	3079	226	462	1035	142	69	6056
		14.55%	2.68%	50.84%	3.73%	7.63%	17.09%	2.34%	1.14%	100

Fig. 4. Labelling Metrics for each Category

within 2-3 percent of each other. The sole exception to this is category 5, the "Who" category. These categories were for README sections that contained information such as: project team, community, mailing list, contact, acknowledgement, licence, and code of conduct. The most likely cause for this 5 percent discrepancy could be accredited to the fact that the categories were manually labeled, and there is bound to be a certain degree of human bias that could account for this difference. Another possible cause of this discrepancy could be attributed to the fact that the original study manually labeled around four times as many README sections. If more data is looked at, it is more likely that the distribution percentage would converge on its true value.

5.3 How was data preprocessed?

5.3.1 *Approach.*

Preprocessing. The first step we had to do in preprocessing the data was to remove stop words and tokenize each section heading and its content. In order to achieve this, we leveraged our knowledge of Databricks and what we did in our assignments and quizzes. A Python UDF was created that handled the two functionalities of removing stop words and tokenizing the data. In order to remove stop words we used NLTK's English stopwords library. And, apart from that we also had a custom list of stopwords and punctuations that we made sure to remove.

The next step in preprocessing was to count how many sections each repository's README file has and check whether any of the section headings have the repository name as part of it. Once again, using Python in Databricks, we wrote the code that collectively gave us the necessary results. Firstly, we had to read each Github repository's URL and parse out just the name of the repository. Next was to loop through each repository's section headings and count how many headings does a repository have and get the section headings as a list of string values. The last two steps involved processing whether a Github README's section has a single-word non-English heading and if there is any non-ASCII text in any of the headings.

During the preprocessing, we also manually calculated the TF-IDF. This involved taking the preprocessed text above and applying a series of transformations. These steps are shown in the results section. We will first calculate the term frequency, which is a count of how often each word occurs in the dataset. Next, we will calculate the document frequency, which is the number of documents, or in our case readme sections, that contain any given word. Once we have both of these, we can calculate the TF-IDF score.

5.3.2 Results. Here, we will take a look at what kind of output the preprocessing steps provided us with. The UDF for removing stopwords and tokenizing can be seen below.

```
    @udf("string")
def preprocess(text):
```

```
from nltk.corpus import stopwords
text = str(text)
words = []
en_stops = stopwords.words('english')
my_stop_words = ["!",'\"',"#","%","$","'","(",")","+","-",".","-",'/',"//","?",
                 "{","}","~","`","_","^","@","<",">",":",";",",","--","=",
                 "/td","/script","href=","class=","divclass=","/li","/div",
                 "/a","1","td","id=","name=","inputtype=","console.log",
                 "true","false","divid=","br","li","else","\"","tr","br/",
                 "/","document.getElementById","name","none",".attr","value=",
                 "2","/span","I","br","><","]","[","*","...","..","``",
                "''","&","Z","X","Y","0",
                "A","B","C","D","E","F","G","x","b","c",
                "/artifactId","................","..."]
en_stops += my_stop_words
sentence = nltk.wordpunct_tokenize(text)
for s in sentence:
  for word in nltk.wordpunct_tokenize(s.lower()):
    if word in en_stops: continue
    if word[0] not in ascii_lowercase: continue
    if len(word) < 3: continue
    words.append(word)
return " ".join(words)
```

Listing 3. Remove Stopwords and Tokenize

The processed data outputted in Databricks can be seen below where all the punctuations and stopwords have been removed along with data being tokenized.

	ProcessedHeading
1	merb
2	modules
3	rubinius language platform
4	code conduct
5	issues support
6	contributing
7	license

Truncated results, showing first 1000 rows.

	ProcessedContent
1	merb web development framework fast simple powerful development information check project website git repository development branch expect things break report issues lighthouse tracker
2	merb consists following parts merb core core functionality needed bootstrap run application merb action args plugin handle request parameters parameters controller actions merb assets asset handling merb slices slices merb application modules mer cache support content caches memcache merb gen generators applications plugins components merb haml support haml merb helpers helpers text html formatting merb mailer mvc mail sending application merb param protection filter request parameters controll...

Fig. 5. Processed Heading and Content

The code snippet below and the subsequent databricks output can be seen below that shows the counting of section headings. And it also shows how different section headings of a Repository are saved in a list.

```
for x in range(1002):
  temp = []
  if result[x] is not None:
    temp.append(result[x])
    headingsAdded = []
    while(True):
      if x < 1002:
        headingsAdded.append(headings[x])
        x = x+1
        count = count+1
        if result[x] is None:
          headingsAdded.append(headings[x])
          continue
        else:
          headingsAdded.pop()
          temp.append(count)
          temp.append(headingsAdded)
          finalMap.append(temp)
          break
  count = 0
```

Listing 4. Count section headings and save in a list

	_1	_2	_3
1	merb	2	["# Merb", "## Modules"]
2	rubinius	18	["# The Rubinius Language Platform\n", "## Code of Conc Support", "## Contributing\n", "## Contributing\n", "## Licen: "## Philosophy & Architecture", "## Philosophy & Architectu Collector", "### Heaps & Garbage Collector", "### CodeDB "### Debugger", "### Profiler", "### Profiler", "### Diagnos Compiler", "### Machine-code Compiler", "### Data Types API", "## FAQ"]
3	exceptionlogger	2	["ExceptionLogger", "CREDITS"]
4	restfulauthentication	12	["# \"Restful Authentication Generator\":http://github.com/t Tracker", "## Documentation", "## Documentation", "## Exc Stories", "### Modularize to match security design patterns: "### Other", "## Non-backwards compatible Changes", "## Passwords\n", "### Validations", "### Validations", "## Insta

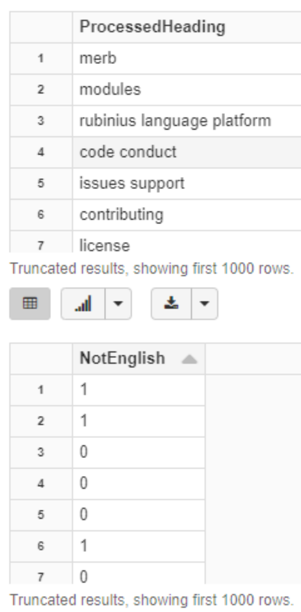
Fig. 6. Section Count and Lists

In figure above, the first column identifies the name of the repository. The second column is the number of sections that we found and manually labelled. The third column has a list of strings for each section's heading. Now, using this Dataframe, we loop through it to find whether a section's heading contains the name of the repository. If the name is part of it we add a 1 to the row and if its not, we add 0. The code for processing non-English single-word headings can be seen below and the output can be seen in the image subsequently posted after that.

```
@udf(returnType=Types.IntegerType())
def notEnglishHeading(text):
  from nltk.tokenize import word_tokenize
```

```
print("j" in words.words())
result = []
text = str(text)
tokenized = nltk.word_tokenize(text)
if(len(tokenized) == 1):
  for s in tokenized:
    if s not in words.words():
      return 1
    else:
      return 0
else:
  return 0
```

Listing 5. Non-English Single Word Heading



	ProcessedHeading
1	merb
2	modules
3	rubinius language platform
4	code conduct
5	issues support
6	contributing
7	license

Truncated results, showing first 1000 rows.

	NotEnglish
1	1
2	1
3	0
4	0
5	0
6	1
7	0

Truncated results, showing first 1000 rows.

Fig. 7. Non-English Headings Classification

In the output dataframe above, a 0 means its not a non-English single-word heading while 1 means that it is a non-English single-word heading.

In the code snippet below, we can see how non-Ascii text in headings is identified. In the image after the code snippet shows the output containing binary labelling for each sections heading.

```
@udf(returnType=Types.IntegerType())
def preprocessNonAscii(text):
  text = str(text)
  words = []
  sentence = nltk.wordpunct_tokenize(text)
  for s in sentence:
    for word in nltk.wordpunct_tokenize(s.lower()):
      if word[0] in ascii_lowercase: return 1
      else: return 0
```

Listing 6. Non-Ascii Text in Headings

	Heading	ProcessedHeading
1	# Merb	0
2	## Modules	0
3	# The Rubinius Language Platform	0
4	## Code of Conduct	0
5	## Issues & Support	0
6	## Contributing	0
7	## License	0

Truncated results, showing first 1000 rows.

Fig. 8. Non-Ascii Headings Classification

Term Frequency. The first step to calculating the TF-IDF is to take the preprocessed text and convert it into a Resilient Distributed Dataset. We can then calculate the term frequency by applying a flat map by values, and then counting the word by value. A snippet of this code is shown below.

```
rdd = df_content_processed.rdd
term_frequency = rdd.flatMapValues(lambda x: word_tokenize(x)).countByValue()
display(term_frequency)
```

Listing 7. Term Frequency

```
defaultdict(int,
            {('1', 'Merb'): 1,
             ('1', 'web'): 1,
             ('1', 'development'): 3,
             ('1', 'framework'): 1,
             ('1', 'fast'): 1,
             ('1', 'simple'): 1,
             ('1', 'powerful'): 1,
             ('1', 'For'): 1,
             ('1', 'information'): 1,
             ('1', 'check'): 1,
             ('1', 'The'): 2,
             ('1', 'project'): 1,
             ('1', 'website'): 1,
             ('1', 'git'): 1,
             ('1', 'repository'): 1,
             ('1', 'This'): 1,
             ('1', 'branch'): 1,
             ('1', 'Expect'): 1,
             ('1', 'things'): 1,
             ('1', 'break'): 1,
Command took 3.78 seconds -- by makissin@ucalgary.ca
```

Fig. 9. Term Frequency Results

Document Frequency. The next step is to calculate document frequency. First we will take each word in the processed text as a dictionary, so that each word is mapped as a key, then tokenize each word. After this we will filter and map each distinct word. Lastly, we apply a count by key. The code to calculating this and the results are shown below.

```
document_frequency = rdd.distinct().filter(lambda x: x[1] != '').map(lambda x:
    word_tokenize(str(x.asDict()["ProcessedContent"]))).countByKey()
display(document_frequency)
```

Listing 8. Document Frequency

```
▸ (1) Spark Jobs
defaultdict(int,
            {'Merb': 3,
             'Rubinius': 4,
             'Participation': 1,
             'Please': 9,
             'welcome': 1,
             'All': 1,
             'install': 7,
             'The': 75,
             'There': 15,
             'Jamis': 1,
             'This': 62,
             'None': 8,
             'Authentication': 1,
             'Added': 2,
             'Here': 7,
             'default': 7,
             'attachment_fu': 2,
             'For': 8,
             'Fields': 1,
             'You': 40,
Command took 4.06 seconds -- by makissin@ucalgary
```

Fig. 10. Document Frequency Results

TF-IDF. Finally, we will have to calculate the actual TF-IDF score using the term frequency and document frequency we just calculated. The function for calculating the TF-IDF score, along with an excerpt of the results is shown below:

```
import numpy as np
from __future__ import division
def tf_idf(N, tf, df):
    result = []
    for key, value in tf.items():
        id = key[0]
        word = key[1]
        df = document_frequency[word]
        if (df>0):
          tf_idf = float(value)*np.log(N/df)

        result.append({"ID":id, "word":word, "score":tf_idf})
    return result
tf_idf_output = tf_idf(N, term_frequency, document_frequency)
tf_idf_output[:10]
```

Listing 9. Document Frequency

```
Out[17]: [{'ID': '1', 'word': 'Merb', 'score': 5.8289456176102075},
 {'ID': '1', 'word': 'web', 'score': 5.8289456176102075},
 {'ID': '1', 'word': 'development', 'score': 5.8289456176102075},
 {'ID': '1', 'word': 'framework', 'score': 5.8289456176102075},
 {'ID': '1', 'word': 'fast', 'score': 5.8289456176102075},
 {'ID': '1', 'word': 'simple', 'score': 5.8289456176102075},
 {'ID': '1', 'word': 'powerful', 'score': 5.8289456176102075},
 {'ID': '1', 'word': 'For', 'score': 4.848116364598481},
 {'ID': '1', 'word': 'information', 'score': 4.848116364598481},
 {'ID': '1', 'word': 'check', 'score': 4.848116364598481}]

Command took 0.07 seconds -- by makissin@ucalgary.ca at 2021-12-15, 11:22:40 AM on
```

Fig. 11. Document Frequency Results

5.4 How do the models perform on original data vs. new + original data?

5.4.1 Approach. From the preprocessed headings and contents of each README file section, three csv files were created containing the category classifications, the preprocessed headings, and the preprocessed contents of each README file section. A single csv file was created for the original dataset, the new dataset, and the combined dataset. The below figure shows a sample of the file created for the combined dataset.

	A	B	C	D	E	F	G	H	I	J	K	L
1	SectionID	ProcessedHeading	ProcessedContent	What	Why	How	When	Who	References	Contribution	Other	
2	1	symon abstr_number s	version abstr_number	1	0	0	1	0	0	0	0	
3	2	abstr_number abstr_nu	symon general purpos	1	0	0	1	0	1	0	0	
4	3	abstr_number abstr_nu	java abstr_number abs	0	0	1	0	0	0	0	0	
5	4	abstr_number abstr_nu	symon simulate multip	1	0	0	0	0	0	0	0	
6	5	abstr_number abstr_number memory maps		1	0	0	0	0	0	0	0	
7	6	abstr_number abstr_nu	abstr_number abstr_n	1	0	0	0	0	0	0	0	
8	7	abstr_number abstr_nu	abstr_number dfff abs	1	0	0	0	0	0	0	0	
9	8	abstr_number abstr_nu	abstr_number ffff abst	1	0	0	0	0	0	0	0	
10	9	abstr_number abstr_nu	abstr_image main win	1	0	0	0	0	0	0	0	
11	10	abstr_number abstr_nu	abstr_image symon lo	1	0	0	0	0	0	0	0	
12	11	abstr_number abstr_nu	abstr_image memory	1	0	0	0	0	0	0	0	
13	12	abstr_number abstr_nu	abstr_image last abstr	1	0	0	0	0	0	0	0	
14	13	abstr_number abstr_nu	abstr_image simulatec	1	0	0	0	0	0	0	0	
15	14	abstr_number abstr_nu	abstr_image breakpoir	1	0	0	0	0	0	0	0	
16	15	abstr_number abstr_nu	abstr_image feature h	1	0	0	0	0	1	0	0	
17	16	abstr_number abstr_nu	program fill video scre	1	0	0	0	0	0	0	0	
18	17	abstr_number abstr_number usage		0	0	1	0	0	0	0	0	
19	18	abstr_number abstr_nu	build symon apache m	0	0	1	0	0	0	0	0	
20	19	abstr_number abstr_nu	simulator requires rom	0	0	1	0	0	0	0	0	
21	20	abstr_number abstr_nu	addition rom images	0	0	1	0	0	0	0	0	
22	21	abstr_number abstr_nu	loading program rom i	0	0	1	0	0	0	0	0	
23	22	abstr_number abstr_nu	abstr_number abstr_n	0	0	0	1	0	0	0	0	
24	23	abstr_number abstr_nu	abstr_number abstr_n	0	0	0	1	0	0	0	0	
25	24	abstr_number abstr_nu	feedback form dialogs	0	0	0	1	0	0	0	0	
26	25	abstr_number abstr_nu	copyright abstr_numbe	0	0	0	0	1	0	0	0	

Fig. 12. Sample Input CSV File

These input files were passed through a pipeline consisting of CountVectorizer, Tokenizer, IDF, and StringIndexer steps, resulting in Pyspark dataframes consisting of the label and the features of each section.

	label	features
1	1	{"vectorType": "sparse", "length": 15264, "indices": [0, 95, 384, 660, 2236, 2458, 13679], "values": [8.632652893577315, 3.7539337620219593, 4.8910123315178655, 5.1592763181125445, 6.471462707078714, 6.625613386905972, 7.724225675574081]}
2	1	{"vectorType": "sparse", "length": 15264, "indices": [0, 114, 214, 372, 573, 589, 728, 992, 1783, 2061, 2294, 2458, 2803, 3144, 7323, 13030, 15237], "values": [3.453061157430926, 3.9287364864018866, 4.5461718452261355, 4.920865294667546, 4.8910123315178655, 5.0500770261475525, 5.198497031265826, 5.527001098237862, 6.471462707078714, 6.471462707078714, 6.625613386905972, 6.625613386905972, 6.625613386905972, 7.031078495014136, 7.724225675574081, 7.724225675574081, 7.724225675574081]}
3	0	{"vectorType": "sparse", "length": 15264, "indices": [0, 51, 194, 264, 500, 601, 6160, 7540], "values": [12.08571405100824, 3.43376623442569, 4.306498991960715, 4.392021165398877, 4.98338565164888, 15.150231078442658, 7.318760567465917, 7.724225675574081]}
4	1	{"vectorType": "sparse", "length": 15264, "indices": [0, 114, 172, 1004, 2064, 3763, 9638], "values": [5.179591736146389, 3.9287364864018866, 4.2126802367430605, 5.527001098237862, 6.625613386905972, 7.031078495014136, 7.724225675574081]}
5	1	{"vectorType": "sparse", "length": 15264, "indices": [0, 77, 123, 149, 205, 308, 320, 411, 1042, 1652, 1900, 1923, 2458, 3144, 3891, 4545, 4567, 5786, 5982, 9266], "values": [29.35101983816287, 3.698873984838932, 3.9175631858037616, 3.951464737479443, 4.588731459644931, 4.9516369533343, 5.085168345958823, 14.673036994553597, 5.527001098237862, 6.471462707078714, 7.031078495014136, 6.220148278797807, 6.625613386905972, 14.062156990028273, 7.031078495014136, 7.031078495014136, 7.031078495014136, 7.318760567465917,

Fig. 13. Sample Preprocessed Dataframe

Using Databricks and Pyspark ML Library, the two columns within these preprocessed dataframes were used in the training process for a selection of models which include:

- LogisticRegression, with hyperparameter tuning of 'maxIter', parameter values of default, 10, 20, 30
- LinearSVC, with hyperparameter tuning of 'maxIter', parameter values of default, 10, 20, 30
- RandomForestClassifier, with hyperparameter tuning of 'numTrees', parameter values of 10, 20, 30
- NaiveBayes, with hyperparameter tuning of 'model', parameter values of multinomial, complement, gaussian

In order to evaluate the performances of the models, a split was performed on the datasets consisting of 25% for the training set and 75% for the validation set. The models were then fit to the training set as demonstrated in the following code block.

```
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.classification import LinearSVC
from pyspark.ml.classification import NaiveBayes
from pyspark.ml.classification import RandomForestClassifier
from pyspark.sql import Row
from pyspark.ml.linalg import Vectors
from pyspark.mllib.evaluation import MulticlassMetrics

# We ran through this notebook for all the hyperparameters that we chose for each
    of the 8 categories individually and independently

lr = LogisticRegression(featuresCol='features',labelCol='label',maxIter=30,
    fitIntercept=True)
svc = LinearSVC(maxIter=30)
rf = RandomForestClassifier(labelCol="label", featuresCol="features", numTrees=30)
nb = NaiveBayes(smoothing=1.0, modelType="gaussian")

train,test = clean_df.randomSplit([0.25, 0.75])
model = lr.fit(train)
predictionsLR = model.transform(test)
model = svc.fit(train)
predictionsSVC = model.transform(test)
model = rf.fit(train)
predictionsRF = model.transform(test)
model = nb.fit(train)
predictionsNB = model.transform(test)
```

Listing 10. Train/Test Split and Model Fitting

For each trained model and set of hyperparameters, the accuracy, precision, recall and F1 scores were calculated using the MulticlassClassificationEvaluator available through Pyspark ML Lib and displayed using the following code block.

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

evaluator = MulticlassClassificationEvaluator()
```

```
print("Logistic Regression Accuracy: " + str(evaluator.evaluate(predictionsLR, {
    evaluator.metricName: "accuracy"})))
print("SVC Accuracy: " + str(evaluator.evaluate(predictionsSVC, {evaluator.
    metricName: "accuracy"})))
print("Random Forest Accuracy: " + str(evaluator.evaluate(predictionsRF, {
    evaluator.metricName: "accuracy"})))
print("Naive Bayes Accuracy: " + str(evaluator.evaluate(predictionsNB, {evaluator.
    metricName: "accuracy"})))
print("-----------------------------------------")
evaluator = MulticlassClassificationEvaluator().setLabelCol("label").
    setPredictionCol("prediction").setMetricName("precisionByLabel")
print("Logistic Regression Precision: " + str(evaluator.evaluate(predictionsLR)))
print("SVC Precision: " + str(evaluator.evaluate(predictionsSVC)))
print("Random Forest Precision: " + str(evaluator.evaluate(predictionsRF)))
print("Naive Bayes Precision: " + str(evaluator.evaluate(predictionsNB)))

print("-----------------------------------------")
evaluator = MulticlassClassificationEvaluator().setLabelCol("label").
    setPredictionCol("prediction").setMetricName("recallByLabel")
print("Logistic Regression Recall: " + str(evaluator.evaluate(predictionsLR)))
print("SVC Recall: " + str(evaluator.evaluate(predictionsSVC)))
print("Random Forest Recall: " + str(evaluator.evaluate(predictionsRF)))
print("Naive Bayes Recall: " + str(evaluator.evaluate(predictionsNB)))

print("-----------------------------------------")
evaluator = MulticlassClassificationEvaluator().setLabelCol("label").
    setPredictionCol("prediction").setMetricName("f1")
print("Logistic Regression F1: " + str(evaluator.evaluate(predictionsLR)))
print("SVC F1: " + str(evaluator.evaluate(predictionsSVC)))
print("Random Forest F1: " + str(evaluator.evaluate(predictionsRF)))
print("Naive Bayes F1: " + str(evaluator.evaluate(predictionsNB)))
```

Listing 11. Evaluation of Metrics

The results of each algorithm and hyperparameter set were collected and compiled into a spreadsheet document. This process was repeated for each of the datasets (original, new and combined), each column within the dataset, and each of the model/hyperparameter combinations.

5.4.2 Results. Using accuracy as an initial metric, the best performing model and set of hyperparameters on the original dataset was found to be the SVC with a maxIter parameter of 30. For the combined dataset, that consists of the original and the newly labelled dataset, the best performing model was found to be the SVC with the default maxIter parameter of 100.

For each model, confusion matrices were generated for each category. Attached in Appendix D, the matrices for an SVC model with 30 iterations and default number of iterations can be found for the original and combined datasets, respectively.

The accuracy, precision, recall, and F1 score across all of these confusion matrices can be summarized by taking the average of the values generated using the multiclass classification evaluator previously discussed. For the original dataset's SVC model with 30 iterations, the averages across the categories are:

- Accuracy: 0.8949714286
- Precision: 0.9039571429
- Recall: 0.9314571429
- F1: 0.8781285714

For the combined dataset's SVC model with default parameters, the averages across the categories are:

- Accuracy: 0.8863714286
- Precision: 0.8932571429
- Recall: 0.9235857143
- F1: 0.8680571429

Comparing the metrics between the original dataset and the combined dataset, we can see a slight decrease in the performance of the combined dataset versus that of the original dataset. However, the difference in performance between the two is relatively minor and not significant, suggesting that both are relatively comparable in terms of performance.

5.5 How do the performance of the models change based on the choice of hyperparameters?

5.5.1 Approach. As mentioned in the previous section, different combinations of datasets, categories, models, and hyperparameters were tested, with their accuracy, precision, recall, and F1 scores measured and placed into a spreadsheet. The results presented in the previous section belonged to the model and hyperparameter set that resulted in the best performance in terms of accuracy.

5.5.2 Results. For each model and hyperparameter set, the average of each metric was taken across the different categories and are summarized in the following tables.

In general, it was found that Logistic Regression and SVC algorithms performed well on all metrics, usually having a score of 0.85 or higher. Random Forest was also a strong contender and performed comparably to the previous two in all metrics except F1, in which it had scores around 0.80.

For accuracy, recall, and F1 metrics, Naive Bayes was found to consistently perform the worst out of the four algorithms aside from one hyperparameter exception. When the model parameter was set to gaussian, the algorithm was found to perform significantly better with scores around 0.80 and higher. When set to multinomial and complement, it tended to fair much worse, with scores in the 0.6 to 0.7 range.

However, when using precision as a metric, Naive Bayes performed quite well, in some cases, even outperforming the other algorithms. This suggests that if precision is of the highest priority, Naive Bayes may be a suitable algorithm.

Algorithm	**Parameters**	**Average Accuracy**
Logistic Regression	default	0.8635
Logistic Regression	maxIter = 10	0.8925857143
Logistic Regression	maxIter = 20	0.8897142857
Logistic Regression	maxIter = 30	0.8927857143
SVC	default	0.8683857143
SVC	maxIter = 10	0.8940142857
SVC	maxIter = 20	0.8913142857
SVC	maxIter = 30	0.8949714286
Random Forest	10 tress	0.8384428571
Random Forest	20 trees	0.8643714286
Random Forest	30 trees	0.8635142857
Naive Bayes	model = multinomial	0.6680571429
Naive Bayes	model = complement	0.6325714286
Naive Bayes	model = gaussian	0.8172571429

Table 3. Hyperparameter Tuning - Original Dataset Accuracy

Algorithm	Parameters	Average Accuracy
Logistic Regression	default	0.8829571429
Logistic Regression	maxIter = 10	0.8767142857
Logistic Regression	maxIter = 20	0.8754428571
Logistic Regression	maxIter = 30	0.878
SVC	default	0.8863714286
SVC	maxIter = 10	0.8797142857
SVC	maxIter = 20	0.8806714286
SVC	maxIter = 30	0.8815142857
Random Forest	10 tress	0.8618285714
Random Forest	20 trees	0.8588285714
Random Forest	30 trees	0.8611428571
Naive Bayes	model = multinomial	0.6696
Naive Bayes	model = complement	0.6386857143
Naive Bayes	model = gaussian	0.7876285714

Table 4. Hyperparameter Tuning - Combined Dataset Accuracy

Algorithm	Parameters	Average Precision
Logistic Regression	default	0.8760857143
Logistic Regression	maxIter = 10	0.9012285714
Logistic Regression	maxIter = 20	0.8965857143
Logistic Regression	maxIter = 30	0.9022714286
SVC	default	0.8791428571
SVC	maxIter = 10	0.8998857143
SVC	maxIter = 20	0.8987571429
SVC	maxIter = 30	0.9039571429
Random Forest	10 tress	0.8768571429
Random Forest	20 trees	0.8957857143
Random Forest	30 trees	0.9216857143
Naive Bayes	model = multinomial	0.8985142857
Naive Bayes	model = complement	0.9196142857
Naive Bayes	model = gaussian	0.8837

Table 5. Hyperparameter Tuning - Original Dataset Precision

Algorithm	Parameters	Average Precision
Logistic Regression	default	0.8890428571
Logistic Regression	maxIter = 10	0.8834428571
Logistic Regression	maxIter = 20	0.8891857143
Logistic Regression	maxIter = 30	0.8917857143
SVC	default	0.8932571429
SVC	maxIter = 10	0.8838571429
SVC	maxIter = 20	0.8907
SVC	maxIter = 30	0.8928142857
Random Forest	10 tress	0.8782714286
Random Forest	20 trees	0.8716
Random Forest	30 trees	0.9215714286
Naive Bayes	model = multinomial	0.8984285714
Naive Bayes	model = complement	0.8949285714
Naive Bayes	model = gaussian	0.8680571429

Table 6. Hyperparameter Tuning - Combined Dataset Precision

Algorithm	Parameters	Average Precision
Logistic Regression	default	0.9109285714
Logistic Regression	maxIter = 10	0.9263
Logistic Regression	maxIter = 20	0.9253857143
Logistic Regression	maxIter = 30	0.9271285714
SVC	default	0.9208571429
SVC	maxIter = 10	0.9319285714
SVC	maxIter = 20	0.9256571429
SVC	maxIter = 30	0.9314571429
Random Forest	10 tress	0.8606714286
Random Forest	20 trees	0.8644285714
Random Forest	30 trees	0.8623428571
Naive Bayes	model = multinomial	0.6701857143
Naive Bayes	model = complement	0.6179857143
Naive Bayes	model = gaussian	0.8568857143

Table 7. Hyperparameter Tuning - Original Dataset Recall

Algorithm	Parameters	Average Recall
Logistic Regression	default	0.9211
Logistic Regression	maxIter = 10	0.9134142857
Logistic Regression	maxIter = 20	0.9053428571
Logistic Regression	maxIter = 30	0.9067142857
SVC	default	0.9235857143
SVC	maxIter = 10	0.9205142857
SVC	maxIter = 20	0.9145571429
SVC	maxIter = 30	0.9126857143
Random Forest	10 tress	0.8599571429
Random Forest	20 trees	0.8571428571
Random Forest	30 trees	0.8573285714
Naive Bayes	model = multinomial	0.6727714286
Naive Bayes	model = complement	0.6385428571
Naive Bayes	model = gaussian	0.8275428571

Table 8. Hyperparameter Tuning - Combined Dataset Recall

Algorithm	Parameters	Average F1
Logistic Regression	default	0.8437285714
Logistic Regression	maxIter = 10	0.8757714286
Logistic Regression	maxIter = 20	0.8734142857
Logistic Regression	maxIter = 30	0.8781428571
SVC	default	0.8463428571
SVC	maxIter = 10	0.8766
SVC	maxIter = 20	0.8740142857
SVC	maxIter = 30	0.8781285714
Random Forest	10 tress	0.7722285714
Random Forest	20 trees	0.8106285714
Random Forest	30 trees	0.8082714286
Naive Bayes	model = multinomial	0.7261285714
Naive Bayes	model = complement	0.7091857143
Naive Bayes	model = gaussian	0.8321142857

Table 9. Hyperparameter Tuning - Original Dataset F1

Algorithm	**Parameters**	**Average F1**
Logistic Regression	default	0.8655714286
Logistic Regression	maxIter = 10	0.8549142857
Logistic Regression	maxIter = 20	0.8595285714
Logistic Regression	maxIter = 30	0.8614571429
SVC	default	0.8680571429
SVC	maxIter = 10	0.8571571429
SVC	maxIter = 20	0.8620571429
SVC	maxIter = 30	0.8627571429
Random Forest	10 tress	0.8052857143
Random Forest	20 trees	0.8004
Random Forest	30 trees	0.8031428571
Naive Bayes	model = multinomial	0.7347428571
Naive Bayes	model = complement	0.7098428571
Naive Bayes	model = gaussian	0.8061285714

Table 10. Hyperparameter Tuning - Combined Dataset F1

5.6 How are misclassifications of the best performing model distributed?

5.6.1 Approach.

What/Why, How, and When. We will first look at the misclassifications for the What/Why, the How and the When category. The what/why category had a total of 623 misclassified sections. The how category had a total of 951 misclassified sections. The when category has a total of 147 misclassified sections. Label of 1 means it is a part of the category and 0 means its not part of the category.

ID	Heading	Content	Label	Pred	Reason
99	submitting lab	submit solution typing learn submit terminal	1	0	The term "solution" may have made it think of this as a what or why category.
464	features	limited subset auto layout anything nslayoutconstraint	0	1	The short length could've been the reason for this missclassification
497	caching	simplified caching using spring new cacheable cache	0	1	The term "Simplified caching" could've lead to the misclassification
2162	angular token authentication	abstr$_{i}$*mage*	0	1	Since this section only has an image and no words it was misclassified
5055	workflow new mailing list	got mailing list talk workflow good come visit post problems ideas anything groups google group ruby workflow see	0	1	The word "workflow"could've led to the misclassification
5149	owncloud cookbook	installs owncloud owncloud org	1	0	The short length and lack of definitive words could've been the reason for this misclassification
5181	philip php irc bot framework	philip slim inspired framwork creating simple irc bots written bill israel purpose project allow people create fun simple irc bots minimal overhead complexity	1	0	The name "philip" may have made the classifier think this is a who category
5204	hubot scripts	collection community scripts hubot chat bot company	1	0	There are no word that definitely sound like a how statement, so out of context this is likely the reason for the misclassification

Table 11. How Category Misclassifications

ID	Heading	Content	Label	Pred	Reason
29	introduction inheritance	real world	1	0	The short length and lack of definitive words could've been the reason for this missclassification
29	introduction	following exercises completed web developer candidates applying position jaguar design studio need use git bitbucket account complete submit exercises expected applicants previous experience git bitbucket successfully using tools part exercise follow instructions provided job posting details complete exercise	1	0	From reading the preprocessed text, it sounds like. it could possibly be a why statement, and this could've been the cause for the misclassification
400	returns	array filtered resulting array	0	1	The short length and lack of definitive words could've been the reason for this missclassification
595	node	node platform built chrome javascript runtime easily building fast	1	0	The lack of any definitive "what" words is likely the reason for the misclassification
4125	fake function inject assistant	fake inject assistant tool fake function replacement unit test easier replace dependency test double	1	0	The lack of any definitive "what" words is likely the reason for the misclassification
4138	dragonet	universal minecraft server	1	0	The short length and lack of definitive words could've been the reason for this missclassification
4210	define job types	whenever ships three pre defined job types command	0	1	The short length and lack of definitive words could've been the reason for this missclassification
4245	stories	cucumber wiki github aslakhellesoy cucumber home features allow expressive enjoyable tests authentication code flexible code resource testing stories extended ben mabey benmabey rspec plain text stories webrat chunky bacon	1	0	The number of strange words, such a cucumber, and other non english words is likely the reason for the misclassification
4280	thin	small fast ruby web server	1	0	The short length and lack of real words could've been the reason for this misclassification.

Table 12. What Category Misclassifications

ID	Heading	Content	Label	Pred	Reason
223	versioning	transparency insight release cycle	1	0	The short length and lack of definitive words could've been the reason for this missclassification
502	note	project much still work progress cli beta wish collaborate project still young	1	0	The most common word is the word "project" which isn't usually associated with "When" this could have been the cause for misclassification
4462	readme	merb core new branch merb also referred merb next series aims provide stable stripped api future merb release branch based release series significant rewrites goals release stabilize public interface methods provide consistent application development experience remove features nothing except central application api left improve comments methods using standard documentation methodology described documentation standards separate tests two sections private public public methods methods tagged public part standard stable merb api private methods implementation methods might implement new render api build extensions regain selected features needed familiarize merb core application might look reference sample directory	1	0	The lack of any definitive "when" type words is likely the reason for misclassification
4465	important	ruby gem longer supported several years longer recommended public use instead would recommend looking officially supported amazon ruby sdk provided amazon see aws amazon sdkforruby thanks support	1	0	The includsion of non-english words such as "sdkforruby" and "sdk" were likely the cause for misclassification
4559	subversion	preparing release braid support subversion repositories removed active maintainers inadequate test coverage anyone motivated add maintain subversion support please contact authors	1	0	The word "authors" which is typically associated with the "Who" or "contributors" category is the likely reason for the misclassification

Table 13. When Category Misclassifications

Who and References. For the Who and References categories, misclassifications tended to occur due to specific words or abstractions causing an incorrect classification. In some cases, there was not enough information for the algorithm to process, resulting in improper labelling. There were also several cases where the algorithm was unable to recognize names as belonging to authors, resulting in a different classification. These misclassifications can be viewed in Appendix B. A sample page of each classification can be found in the following two tables.

ID	Heading	Content	Label	Pred	Reason
25	copyright acknowledgements	copyright seth morabito web loomcom com portions copyright maik merten maikmerten googlemail com additional components used project copyright respective owners enhanced basic copyright lee davison functional tests copyright klaus dormann jterminal copyright graham edgecombe project would possible without following resources hyperlink	1	0	The hyperlinks may have been interpreted as References or Contributions.
84	license	vlayout available mit license	1	0	Vlayout may be interpreted as something unrelated to Who.
88	license	corert repo licensed	1	0	The hyperlinks may have been interpreted as References or Contributions.
89	net foundation		1	0	Since this section only has an image and no words it had insufficient content to classify properly.
109	license author	author david king	1	0	David King may not have been recognized as a name.
111	lastpass	command line interface	1	0	The hyperlink may have been interpreted as References or Contributions.
130	developed	pedro vicente	1	0	The hyperlinks may have been interpreted as References or Contributions. Pedro Vicente may not have been recognized as a name.
195	license authors	author adam jacob author joshua timberman author bryan mclellan author dave esposito author david abdemoulaie author edmund haselwanter author eric rochester author jim browne author matthew kent author nathen harvey author ringo smet author sean omeara author seth chisamore author gilles devaux author sander van zoest author taylor price copyright	1	0	The hyperlinks may have been interpreted as References or Contributions. Names may not have been recognized.

Table 14. Who Category Misclassifications

ID	Heading	Content	Label	Pred	Reason
49	works	simple service takes url returns rendered html script tags removed note proxy request server using middleware relative links css images etc still work get http service prerender https www google com get http service prerender https www google com search angular	0	1	Words like 'url', 'links', and 'google' may have been interpreted as belonging to a References category.
50	running locally	trying test prerender website localhost	0	1	Words like 'website' and 'localhost' may have been interpreted as belonging to a References category.
5106	dbconsole	command line database directly script dbconsole would connected database credentials defined database yml starting script without arguments connect development database passing argument connect different database like script dbconsole production currently works mysql postgresql sqlite	1	0	Multiple occurences of 'database' and related words may have been interpreted as belonging to a What or How classification.
5111	usage	simply run client php commandline optionally passing url atom service document start defaults uri index atom	1	0	Words like 'run' and 'passing' may have been interpreted as a How category.
5120	install	gem install i18n$_data$	0	1	'Gem' may have been interpreted as belonging to a References category.
5126	authors	contributors barry allard richard doe michael grosser michael grosser license mit build status	1	0	The names may have triggered a Who or Contribution classification.
5137	internal methods	dragons warned head2 build setup geo coder object steps construction geo coder new steps google api key abcdefghjijklmnopqrstuvwxyz yahoo appid wharrrrgarrrrgarrrrgarrrrrblllll	1	0	Words like 'step' or 'api' may have triggered a How classification.
5138	author	shirley jshirley toeat devin austin dhoss toeat	1	0	Names may have triggered a Who classification.

Table 15. References Category Misclassifications

Contribution and Other. Now, we will talk about the misclassification done by the model for the Contribution and Other categories. Looking at the data, the Contribution category had 74 data points misclassified. While, for the Other category, it had 27 misclassified data points. The two tables below shows few randomly selected datapoints missclassified in the Contribution category. Classification of 1 means it is Contribution and 0 means its not Contribution and same rules follow for Other category. For rest of the classifications selected in Contribution and Other category, please refer to Appendix 8.3 part C.

ID	Heading	Content	Label	Pred	Reason
134	contributing	encourage contribute ruby rails please check abstr_hyperlink guidelines proceed abstr_hyperlink everyone interacting rails sub projects codebases	1	0	lots of words in content could have made the model decide to not focus on the "contribute" keyword
208	node	abstr_hyperlink abstr_hyperlink node javascript runtime built chrome abstr_number javascript engine node uses event driven	1	0	lots of technical words like javascript, chrome and node that might have led to prediction of 0
230	contributing	please submit pull requests wip branches pull request contains javascript patches features	1	0	the algorithm could have looked at keywords like submit, requests and features to decide its not Contribution
240	contributing	pull requests way apologise advance slow action pull requests issues two rules submitting pull request match naming convention camelcase	1	0	the algorithm could have looked at keywords like submitting, requests to decide its not Contribution
281	issues	find bug want request new feature please let know submitting issue	1	0	keywords like submitting, issues, bug and request could have been major factor
282	contributing	esri welcomes contributions anyone everyone please see abstr_hyperlink	1	0	Difficult to say why model misclassified despite the content and heading having keywords contributing and contribution
308	contributors welcome	questions	1	0	not enough content for model to decide
324	contributing	please refer abstr_hyperlink	1	0	content does not have useful keywords
502	note	project much still work progress cli beta wish collaborate project still young	1	0	keywords like note, collaborate could have lead to misclassification
563	getting involved	want report bug	1	0	keyword like bug, report could be why its misclassified
592	contributing	fork submit pull requests improve tool	1	0	difficult to say why despite having keyword contributing

Table 16. Contribution Category Misclassifications

Next, in the tables below we look at all of the misclassifications for the Other category.

ID	Heading	Content	Label	Pred	Reason
100	hello world history	small piece coding history' handwritten version hello world early programming language abstr_image abstr_hyperlink based abstr_number bell laboratories internal memorandum brian kernighan	1	0	lots of technical terms like hello world, programming, hyperlink, language which could be why it was misclassified
303	replacing several methods instead whole class	calculator class example ruby methods	1	0	lots of technical terms like hello world, programming, hyperlink, language which could be why it was misclassified
355	operations	insert delete retain	1	0	not enough content for the model to make a sound decsision
356	construction	constructor insert delete retain	1	0	not enough content for the model to make a sound decsision
357	documents	methods called non document deltas result undefined behavior concat diff eachline	1	0	not enough quality keywords for the model to make a sound decsision
359	operational transform	compose transform transformposition	1	0	not enough quality keywords for the model to make a sound decsision
425	methods	slice slice start slice start	0	1	keyword like methods could have been why it was misclassified
535	disclaimer	cntk active use microsoft constantly evolving bugs	1	0	bugs and disclaimer keywords could have made the model classify it into another category
784	grading	assignment graded via peer assessment	1	0	unsure why this was misclassified, hard to tell based on limited content
2333	lyla says	app starting shape feels bit quite places put finger feels	1	0	lacks quality content to make a sound decision
2335	kagure says	color scheme really sad feel sad	1	0	lacks quality content to make a sound decision

Table 17. Other Category Misclassifications

5.6.2 Results. From looking through all of the misclassified records we see that the misclassifications typically fall into one of three categories. They are typically sections that are very short, sections that have one or more words that are typically associated with a different category, or the inclusion of non-English words or acronyms that were not removed in the preprocessing.

In the case of the section content being very short, since there are very few words, the model has very little data to go off and will therefore be less likely to make a correct classification.

In the case of there being a word that is typically associated with a different category, the inclusion of that key word would likely confuse the classifier since the word would occur frequently in other categories.

The final case is when there are non-English words or symbols that didn't get remove by the preprocessing. While the preprocessing should remove all stopwords and characters that are not useful to the model, some do slip through. During classification the cases where there are unproccessed words typically are misclassifed since the model does not have any idea how to accurately classify these words.

6 DISCUSSIONS

Meet Pandya's Scenario.
One scenario that I could think of is with related to teachers or tutors in the computer science fields. If they are looking for repositories as examples to discuss with their students, they would like to ensure that the repos they find have well written Readme files. And, this is where our model could be really useful.

Michael Kissinger's Scenario.
The model can be used by software developers who are trying to solve a specific problem, but don't want to spend too much time searching through countless github repositories for the solution. Our model can classify sections of the repository, which the developer can then read through to find what they are looking for. For example, the model could be modified to just show the "what" sections of the read me, which the developer can then search through to determine which repositories will likely contain helpful information.

Brandon Quan's Scenario
The model could be used to help develop a template for README files. Suppose a software developer has come across a repository that they found useful in one of their projects, but has identified a few bugs in the code. By having the README file organized in a way such that the README's sections are categorized by the eight classifications used by the model, all the developer would need to do is refer to the Who or Contributions section of the README file to find out who to contact in order to address the bugs.

7 CONCLUSION

In conclusion, as a group we were able to successfully replicate the work done in the original research paper and apply it to newly fetch 1000 data points. For the new data points, the labelling was done manually by the three members of this team. Once labelling was done, we checked each other's labels for quality purposes. Finally for any disagreement, we had the third labeller make the final decision.

For all of the coding components, we used PySpark with Databricks. The data preprocessing aspect was done successfully to be able to extract and have statistical and heuristic features. Features involved calculating TF-IDF score, check whether a section contains name of the repository, check for non-English single-word headings and non-ASCII text in a section. Finally, for training of the models we used PySpark's ML library that comes with a lot of different functionalities and abilities to use the well-known machine learning algorithms.

We trained the original, new, and combined datasets on SVC, Naive Bayes, Random Forest and Logistic Regression models. For each of the model, different hyperparameter sets were tested to see how the performance changed. Eventually, it was found that SVC with default parameters was the best performing model on the combined dataset and SVC with a maxIter parameter of 30.

One interesting observation was with the Naive Bayes model when using the Gaussian model. It was found to perform quite well specifically when using the Gaussian model, but with Multinomial and Complement models, its performance dropped significantly in all metrics but precision.

The work we did in this project can most definitely be extended and improved. Machine learning is an iterative process so the more iterations are applied the better results we could expect. There are a lot of key factors that make a difference as to how well the model will perform. These include the dataset, labelling, hyperparameters, choice of model, etc.

REFERENCES

[1] Ferdian Thung Thushari Atapattu David Lo Gede Artha Azriadi Prana, Christoph Treude. 2018. *Categorizing the Content of GitHub README Files.* https://arxiv.org/pdf/1802.06997.pdf

[2] Github. [n. d.]. *Github REST API.* Retrieved December 12, 2021 from https://docs.github.com/en/rest

[3] gprana. [n. d.]. *READMEClassifier.* Retrieved December 12, 2021 from https://github.com/gprana/READMEClassifier

8 APPENDIX

8.1 A - What,Why,How and When Misclassifications

ID	Heading	Content	Label	Pred	Reason
99	submitting lab	submit solution typing learn submit terminal	1	0	The term "solution" may have made it think of this as a what or why category".
100	hello world history	small piece coding history handwritten version hello world early programming language abstrimage abstrhyperlink based abstrnumber bell laboratories internal memorandum brian kernighan	0	1	The words such as handwritten may have made it think it is a how section.
101	description	words currently tested abstrnumber bit ubuntu abstrnumber abstrnumber lucid	0	1	The lack of definitive words could've let to this being misclassified.
197	features	supports abstrhyperlink api abstrhyperlink supports interceptors request response supports latest firefox	0	1	The lack of definitive words could've let to this being misclassified.
233	animate	add water css animation animate css bunch cool	0	1	The short length could've been the reason for this misclassification.
463	code snippets	copy included code snippets library developer xcode userdata codesnippets write masonry blocks lightning speed masmake view masmakeconstraints masconstraintmaker make code masupdate view masupdateconstraints masconstraintmaker make code masremake view masremakeconstraints masconstraintmaker make code	1	0	There are several words that are combined which didn't get caught in the preprocessing, this could be the reason for the misclassification.
464	features	limited subset auto layout anything nslayoutconstraint	0	1	The short length could've been the reason for this misclassification.
465	todo	eye candy mac example project tests examples	0	1	The word tests could've lead to the misclassification.
497	caching	simplified caching using spring new cacheable cache	0	1	The term "Simplified caching" could've lead to the misclassification.
498	spring	streamlined configuration web	0	1	The term "configuration" could've led to the misclassification.
499	testing	unit testing junit	0	1	The short length could've been the reason for this missclassification.
500	libraries used	spring abstrnumber abstrnumber	1	0	The short length and lack of definitive words could've been the reason for this misclassification.

Table 18. How Category Misclassifications

ID	Heading	Content	Label	Pred	Reason
1096	gathering mac addresses	according connected address abstrnumber abstrnumber abstrnumber abstrnumber route abstrnumber abstrnumber abstrnumber abstrnumber things get bit tricky need bit technical knowledge ips work wont going full details	1	0	The large number of abstrnumbers may have confused the algorithm .
1097	performing attack	gotten address mac address victim	1	0	There is now word that definitely sounds like they should be placed in the how category.
1098	extra	video performing steps	0	1	The word "performing" may have made the classifier think this is a part of the how category.
2162	angular token authentication	abstrimage	0	1	Since this section only has an image and no words it was misclassified.
2163	objectives	end	0	1	The short length could've been the reason for this misclassification .
5053	acknowledgement sponsor	wrote railscamp adelaide november railscamps awesome conference without conference part coding lightning talks alcohol guitar hero thanks people demo tabtab development thoughts final dsl finally helping give tabtab railscamp kept coding tabtab instead proper work therefore project sponsor mocra premier iphone rails consultancy mocra like tabtab donate money hire next rails iphone party	0	1	The word "wrote" could've been the reason for the misclassification
5179	contact	roman efimov github romaonthego twitter romaonthego romefimov gmail	0	1	There are several words that are either non-english word or two words combined. This likely confused the classifier.
5181	philip php irc bot framework	philip slim inspired framwork creating simple irc bots written bill israel purpose project allow people create fun simple irc bots minimal overhead complexity	1	0	The name "philip" may have made the classifier think this is a who category.

Table 19. How Category Misclassifications

ID	Heading	Content	Label	Pred	Reason
5054	license	mit license copyright nic williams drnicwilliams permission hereby granted free charge person obtaining copy software associated documentation files software deal software without restriction including without limitation rights use copy modify merge publish distribute sublicense sell copies software permit persons software furnished subject following conditions copyright notice permission notice shall included copies substantial portions software software provided without warranty kind express implied including limited warranties merchantability fitness particular purpose noninfringement event shall authors copyright holders liable claim damages liability whether action contract tort otherwise arising connection software use dealings software	1	0	The name "William" could've made this be categorized as a who category.
5055	workflow new mailing list	got mailing list talk workflow good come visit post problems ideas anything groups google group ruby workflow see	0	1	The word "workflow" could've led to the misclassification.
5147	dependencies	project relies jquery jquery scrollto plugin	0	1	The use of jquery may have confused the algorithm and lead to the misclassification.
5149	owncloud cookbook	installs owncloud owncloud org	1	0	The short length and lack of definitive words could've been the reason for this misclassification.
5204	hubot scripts	collection community scripts hubot chat bot company	1	0	There are no word that definitely sound like a how statement, so out of context this is likely the reason for the misclassification.
5205	discovering	check hubot script catalog list description available scripts	1	0	There are no word that definitely sound like a how statement, so out of context this is likely the reason for the misclassification.

Table 20. How Category Misclassifications

ID	Heading	Content	Label	Pred	Reason
5213	description	custom metaboxes fields cmb short create metaboxes custom fields blow mind links github project page documentation github wiki field types text text small text medium text money date picker date picker unix timestamp date time picker combo unix timestamp time picker color picker textarea textarea small textarea code select radio radio inline taxonomy radio taxonomy select checkbox multi-check wysiwyg tinymce image file upload oembed field types github wiki	1	0	From reading the preprocessed test, this sounds like a what category, so this is likely why it was misclassified.
5214	installation	script easy install figure probably using place metabox directory inside activated theme inside themes twentyten lib metabox include init php preferably init wordpress hook see example functions php guidance profit	0	1	The term "install" sounds like it should be a part of the how category, so likely that's the reason for the misclassification.
5218	installation	install module type following perl makefile make make test make install	0	1	The term "install" sounds like it should be a part of the how category, so likely that's the reason for the misclassification.
5219	EM-Proxy	"This module requires the JSON module for its communication, available on CPAN. JSON::XS is recommended for faster JSON communication, but not required. Also, the module requires IO::Socket::UNIX if you want to do communication over UNIX sockets, or IO::Socket::INET if you want to do communication over TCP sockets. This module is tested against JSON 2.53, JSON::XS 2.32, IO::Socket::UNIX 1.23, IO::Socket::INET 1.31 and Perl 5.12.3."	0	1	The term JSON and CPAN could have confused the classifier and lead to misclassification.

Table 21. How Category Misclassifications

ID	Heading	Content	Label	Pred	Reason
9	abstrnumber abstrnumber	serial console cpu status abstrimage main window simulator acts primary input output system virtual serial terminal terminal attached simulated acia	1	0	The lack of any definitive "what" words is likely the reason for the misclassification.
29	introduction inheritance	real world	1	0	The short length and lack of definitive words could've been the reason for this missclassification.
93	objectives	abstrnumber create new ruby file abstrnumber write syntactically valid code produce hello world abstrnumber run ruby file abstrnumber run learn gem abstrnumber submit learn lab	1	0	The large number of "abstrnumber" likely caused the misclassification.
149	apache abstrnumber cookbook	abstrhyperlink abstrhyperlink cookbook provides complete debian ubuntu style apache httpd configuration non debian based distributions red hat centos	1	0	The large number of strange, non-English word is likely the cause for the misclassification.
312	introduction	following exercises completed web developer candidates applying position jaguar design studio need use git bitbucket account complete submit exercises expected applicants	1	0	From reading the preprocessed text, it sounds like. it could possibly be a why statement, and this could've been the cause for the misclassification.
400	returns	array filtered resulting array	0	1	The short length and lack of definitive words could've been the reason for this misclassification.
595	node	node platform built chrome javascript runtime easily building fast	1	0	The lack of any definitive "what" words is likely the reason for the misclassification.
741	swift	abstrnumber abstrnumber abstrcodesection	0	1	The short length and lack of definitive words could've been the reason for this misclassification
1100	ionic native	abstrhyperlink abstrhyperlink ionic native curated set wrappers cordova plugins make adding native functionality need abstrhyperlink	1	0	The large number of "abstrhyperlink" likely caused the misclassification.
4125	fake function	inject assistant fake inject assistant tool fake function replacement unit test easier replace dependancy test double	1	0	The lack of any definitive "what" words is likely the reason for the misclassification.

Table 22. What Category Misclassifications

ID	Heading	Content	Label	Pred	Reason
4138	dragonet	universal minecraft server	1	0	The short length and lack of definitive words could've been the reason for this misclassification.
4157	node taas client	node module interface things service	1	0	The short length and lack of definitive words could've been the reason for this misclassification.
4210	define job types	whenever ships three pre defined job types command	0	1	The short length and lack of definitive words could've been the reason for this misclassification .
4245	stories	cucumber wiki github aslakhellesoy cucumber home features allow expressive enjoyable tests authentication code flexible code resource testing stories extended ben mabey benmabey rspec plain text stories webrat chunky bacon	1	0	The number of strange words, such a cucumber, and other non english words is likely the reason for the misclassification.
4280	thin	small fast ruby web server	1	0	The short length and lack of real words could've been the reason for this misclassification.
4306	credits	resourcecontroller created maintained james golick jamesgolick	0	1	The inclusion of a name in the text is likely the reason for misclassifiation, it maye have determined this to have been a "who" section.
4381	modular classic style	contrary common belief nothing wrong classic style suits application switch modular application main downsides using classic style rather modular style may one sinatra application per ruby process plan use one switch modular style reason cannot mix modular classic style switching one style aware slightly different default settings setting classic modular appfile file loading sinatra file subclassing sinatra base run appfile logging methodoverride inlinetemplates static	0	1	The lack of any definitely words typically associated with "what" may have been the reason for misclassification.

Table 23. What Category Misclassifications

ID	Heading	Content	Label	Pred	Reason
1	symon abstrnumber system simulator	version abstrnumber abstrnumber abstrnumber last updated abstrnumber january	1	0	The large number of "abstrnumber" likely caused the misclassification.
2	abstrnumber abstrnumber	symon general purpose simulator systems based mos technologies	1	0	The includsion of non-english words such as "symon", "mos" and "symon" were likely the cause for misclassification microprocessor compatibles symon implemented java core goals accuracy.
223	versioning	transparency insight release cycle	1	0	The short length and lack of definitive words could've been the reason for this misclassification.
502	note	project much still work progress cli beta wish collaborate project still young	1	0	The most common word is the word "project" which isn't usually associated with "When" this could have been the cause for misclassification.
503	webpack update	changed build system beta abstrnumber beta abstrnumber	1	0	The large number of "abstrnumber" likely caused the misclassification.
763	trust system	historical data carry classification user	1	0	The short length and lack of definitive words could've been the reason for this misclassification.
764	badge system	record achievements users contribution community	1	0	The short length and lack of definitive words could've been the reason for this misclassification.
1744	jquery colour picker tiny colour picker useful extra features	abstrnumber abstrhyperlink	1	0	The short length and lack of real words could've been the reason for this missclassification .

Table 24. When Category Misclassifications

ID	Heading	Content	Label	Pred	Reason
4462	readme	merb core new branch merb also referred merb next series aims provide stable stripped api future merb release branch based release series significant rewrites goals release stabilize public interface methods provide consistent application development experience remove features nothing except central application api left	1	0	The lack of any definitive "when" type words is likely the reason for misclassification.
4465	important	ruby gem longer supported several years longer recommended public use instead would recommend looking officially supported amazon ruby sdk provided amazon see aws amazon sdkforruby thanks support	1	0	The includsion of non-english words such as "sdkforruby" and "sdk" were likely the cause for misclassification.
4492	feature requests	able specify conditional filters	1	0	The short length was likely the reason for this missclassification.
4559	subversion	preparing release braid support subversion repositories removed active maintainers inadequate test coverage anyone motivated add maintain subversion support please contact authors	1	0	The word "authors" which is typically associated with the "Who" or "contributors" category is the likely reason for the misclassification.
4702	example	consider following example specification describe postscontroller describe handling get posts cache page lambda get index cache page posts end cache rss feed lambda get index format rss cache page posts rss end end end cache page matcher tests lambda actually triggers caching describe handling get users cache action lambda get show userid user cacheaction	1	0	The large number of "cache page" likely caused the misclassification.
4835	bilevel	pbm pgm files supported future like support tiff reading writing would also help toward goal creating full pdf output	1	0	The non-english words like pdm and pgm may have caused the misclassification.
4866	templates	generate kinds projects using template option another sinatra app template sinatra another library template far templates ruby sinatra adding soon	1	0	The lack of any definitely words typically associated with "when" may have been the reason for misclassification.

Table 25. When Category Misclassifications

8.2 B - Who and References Misclassifications

ID	Heading	Content	Label	Pred	Reason
208	node	node javascript runtime built chrome javascript engine node uses event driven	1	0	The hyperlinks may have been interpreted as References or Contributions.
217	current project team members	node project team comprises group core collaborators sub group forms technical committee ctc governs project information governance node project	1	0	The word 'project' may have been interpreted as belonging to a What category.
218	ctc core technical committee	anna henningsen anna addaleax net ben noordhuis info bnoordhuis chalkerx gmail com chris dickinson christopher dickinson gmail com colin ihrig cjihrig gmail com evan lucas evanlucas com jeremiah senkpiel fedor indutny fedor indutny gmail com james snell jasnell gmail com michael dawson ibm com julien gilli jgilli nodejs org brian white mscdex mscdex net ali ijaz sheikh ofrobots google com rod vagg rod vagg org shigeki ohtsu ohtsu iij myles borins myles borins gmail com sakthipriyan vairamani thechargingvolcano gmail com trevor norris trev norris gmail com rich trott rtrott gmail com	1	0	The hyperlinks may have been interpreted as References or Contributions. Names may not have been recognized.
220	release team	releases node signed one following gpg keys chris dickinson christopher dickinson gmail com colin ihrig cjihrig gmail com bafa bdbe evan lucas evanlucas com ffd james snell jasnell keybase dcfd ede jeremiah senkpiel fishrock keybase myles borins myles borins gmail com dfff rod vagg rod vagg org bae sam roberts octetcloud keybase full set trusted release keys imported running see section verifying binaries details keys verify downloaded file official previous releases node signed one following gpg keys isaac schlueter izs julien gilli jgilli fastmail timothy fontaine tjfontaine gmail com dfd	1	0	Names may not have been recognized. Numbers may have not been recognized as being phone numbers.

Table 26. Who Category Misclassifications

ID	Heading	Content	Label	Pred	Reason
5164	date slider	django template tag helps generate pagination dates	1	0	'Template' may have been interpreted as belonging to a What category.
5165	usage	add app pypi yet clone repository root project myproject git clone github rmasters date slider git better git submodule add github rmasters date slider git date slider add installed apps ensure bottom override templates installed apps django contrib auth django contrib contenttypes myapp date slider control display template want control load date slider date slider date without overriding built templates produces html like current current date march days included default options todo date item display override inherit template date slider slider html may need alternative template loader using django apptemplates directory structure like version slider html used instead default project date slider templates date slider slider html default slider html app templates date slider slider html slider html way without extra loader please submit issue create overriding inheriting template like note date slider part path specific django apptemplates override blocks defined slider html extends date slider date slider slider html block list open class date slider endblock block current item class current item href date date date endblock block item href date date date endblock date class contains datetime date instance couple helpers	1	0	'Add' and 'clone' may have been interpreted as instructions belonging to How.

Table 27. Who Category Misclassifications

ID	Heading	Content	Label	Pred	Reason
5170	introduction	sphinx source plone cms developer manual used site developer plone org read documentation formatted web browser please head plone developer documentation learn update manage documentation tools read writing updating documentation clarifications source folder contains sphinx manual source src folder target plone source code checkouted source code documentation inclusion uploading documentation plone org longer supported instead readthedocs org preferred method distribution	1	0	'Documentation' may have been interpreted as belonging to References or How.
5172	license	copyright plone foundation individual contributors program free software redistribute modify terms gnu general public license published free software foundation either version license option later version program distributed hope useful without warranty without even implied warranty merchantability fitness particular purpose see gnu general public license details received copy gnu general public license along program write free software foundation inc franklin street fifth floor boston usa	0	1	'License', 'public', and 'copy' may have been interpreted as belonging to a Who classification.

Table 28. Who Category Misclassifications

ID	Heading	Content	Label	Pred	Reason
5177	example usage	remenuitem homeitem remenuitem alloc initwithtitle home subtitle return home screen image uiimage imagenamed icon home highlightedimage nil action remenuitem item nslog item item remenuitem exploreitem remenuitem alloc initwithtitle explore subtitle explore additional options image uiimage imagenamed icon explore highlightedimage nil action remenuitem item nslog item item remenuitem activityitem remenuitem alloc initwithtitle activity subtitle perform additional activities image uiimage imagenamed icon activity highlightedimage nil action remenuitem item nslog item item remenuitem profileitem remenuitem alloc initwithtitle profile image uiimage imagenamed icon profile highlightedimage nil action remenuitem item nslog item item remenu alloc initwithitems homeitem exploreitem activityitem profileitem showfromnavigationcontroller self navigationcontroller	1	0	'Readwrite' may have been interpreted as an instruction in a How category.
5185	api	philip api simple similar javascript event system add functionality bot telling	1	0	Words like 'javascript' and 'bot' may have been interpreted as belonging to a What or How category.

Table 29. Who Category Misclassifications

ID	Heading	Content	Label	Pred	Reason
5186	plugins	philip supports basic plugin system adding plugin bot simple using plugin using plugin simple plugins composer able start include plugins via composer run composer update plugins available bot load plugins calling either loadplugin load one time loadplugins array load multiple plugins example plugin whose full namespaced classname example philip plugin helloplugin load either following bot new philip array bot loadplugin new example philipplugin helloplugin bot bot loadplugins array new example philipplugin helloplugin bot plugins accept second optional parameter constructor plugin requires configuration loading plugin accepts configuration might look like bot new philip array bot loadplugin new example philipplugin helloplugin bot config helloplugin additionally like turn bot functionality plugin easy well writing plugin creating plugin simple plugin must extend philip abstractplugin class must provide implementation init getname method plugin named anything however convention philip plugins named like xxx plugin example simple plugin example philipplugin helloplugin php php namespace example philipplugin use philip abstractplugin baseplugin class helloplugin extends baseplugin heavy lifting initializing plugin behavior public function init bot onchannel hello function event request event getrequest event addresponse response msg request getsource request getsendinguser returns plugin public function getname return helloplugin	1	0	Several words relating to 'plugin' may have been interpreted as belonging to a What or How category.

Table 30. Who Category Misclassifications

ID	Heading	Content	Label	Pred	Reason
5187	license	copyright bill israel bill israel gmail permission hereby granted free charge person obtaining copy software associated documentation files software deal software without restriction including without limitation rights use copy modify merge publish distribute sublicense sell copies software permit persons software furnished subject following conditions copyright notice permission notice shall included copies substantial portions software software provided without warranty kind express implied including limited warranties merchantability fitness particular purpose noninfringement event shall authors copyright holders liable claim damages liability whether action contract tort otherwise arising connection software use dealings software license also included license file	1	0	Words like 'obtaining', 'modify', 'merge' may have been interpreted as instructions belonging to a How category.
5188	author	bill israel github epochblue twitter epochbluee	1	0	Names may not have been recognized.
5204	hubot scripts	collection community scripts hubot chat bot company	0	1	'Community' may have triggered a Who classification.
5210	disclaimer	software package developed maintained scientists noaa fisheries alaska fisheries science center considered fundamental research communication reccomendations conclusions presented authors software construed official communication nmfs noaa dept commerce addition reference trade names imply endorsement national marine fisheries service noaa best efforts made insure highest quality tools constant development subject change	1	0	The word 'software' may have be misinterpreted as belonging to a What or How category, and the word 'reference' may have been interpreted as belonging to a References category.
5218	installation	install module type following perl makefile make make test make install	1	0	The word 'install' may have been misinterpreted as belonging to a How classification.

Table 31. Who Category Misclassifications

ID	Heading	Content	Label	Pred	Reason
2		symon general purpose simulator systems based mos technologies microprocessor compatibles symon implemented java core goals accuracy	1	0	Words such as 'systems', 'microprocessor', and 'java' may have been interpreted as belonging to a What classification.
15	experimental crtc video	feature highly experimental possible open video window view menu window simulates output mos crt controller located address default	1	0	Words such as 'highly', 'experimental', 'view', and 'menu' may have been interpreted as belonging to a What or How category.
29	introduction inheritance	real world	0	1	'Real' and 'world' are the only words in the content and weren't enough for the algorithm to properly predict the correct label.
36	prerender service	google	0	1	'Google' was the only word in the content and wasn't enough for the algorithm to properly predict the correct label.
39	javascript	express	1	0	Not enough content for the algorithm to predict a proper label.
40	ruby	rails	1	0	Not enough content for the algorithm to predict a proper label.
42	nginx		1	0	Not enough content for the algorithm to predict a proper label.
45	java		1	0	Not enough content for the algorithm to predict a proper label.
46	grails		1	0	Not enough content for the algorithm to predict a proper label.
47	nginx		1	0	Not enough content for the algorithm to predict a proper label.

Table 32. References Category Misclassifications

ID	Heading	Content	Label	Pred	Reason
5142	parable little smalltalk	small implementation smalltalk based directly tim budd little smalltalk version reorganized source cleaning making work modern unix like oses bsd linux goals remainder consistent formatting source move documentation manaul txt restructuredtext ensure builds runs linux box	1	0	Words like 'source' or 'documentation' may have been interpreted as belonging to a When or How category.
5153	contributing	todo optional public cookbook detail process contributing private cookbook remove section fork repository github create named feature branch like add component write change write tests change applicable run tests ensuring pass submit pull request using github	1	0	Words like 'todo', 'contributing' may have caused a When or Contribution classification.
5155	bootstrap	bootstrap sleek intuitive powerful front end framework faster easier web development created maintained mark otto jacob thornton get started checkout getbootstrap	1	0	Names and words like 'development' and 'framework' may have triggered a What or How classification.
5161	contributing	please submit pull requests wip branches pull request contains javascript patches features must include relevant unit tests html css conform code guide maintained mark otto thanks	1	0	Words like 'pull requests' may have caused a Contribution classification.
5203	global ignores	git global configuration applies rules projects example git config global core excludesfile global ignore apply rules global ignore repos useful use editor like emacs drops backup files work environment generates binary intermediate files always ignored	1	0	Multiple occurences of 'git' may have triggered a How classification.
5204	hubot scripts	collection community scripts hubot chat bot company	0	1	Words like 'community' and 'company' may have been interpreted as a References category.

Table 33. References Category Misclassifications

ID	Heading	Content	Label	Pred	Reason
5211	example	load packages example library agtrend loading required package coda loading required package matrix loading required package mgcv loading required package nlme mgcv overview type help mgcv package loading required package truncnorm agtrend demo available github nmml agtrend library dplyr attaching package dplyr following object masked package nlme collapse following objects masked package stats filter lag following objects masked package base intersect setdiff setequal union load data included agtrend package filter data want example data wdpsnonpups wdpsnonpups filter year droplevels wdpsnonpups count number positive counts site remove sites positive count wdpsnonpups group by site summarise num counts sum count ungroup nz counts wdpsnonpups left join nz counts filter num counts select num counts mutate site factor site wdpsnonpups joining site add photo method covariate data oblique photos prior surveys wdpsnonpups mutate obl integer year data frame wdpsnonpups next step create prediction availability models site based number surveys trend models nonzero counts constant trend signal nonzero counts linear trend signal nonzero counts rw2 spline trend signal zero inflation avail models surveys constant inflation effect surveys linear inflation effect surveys nonzero counts availability model wdpsmodels wdpsnonpups group by site summarize num surv nz count sum count ungroup mutate trend character cut nz count labels const lin avail character cut num surv labels const lin mutate	1	0	Multiple occurences of package names and words like 'load' and 'example' may have triggered a How classification.

Table 34. References Category Misclassifications

8.3 C - Contribution and Other Misclassifications

ID	Heading	Content	Label	Pred	Reason
613	governance current members	node docker image governed docker working group see abstr_hyperlink learn group structure contributing guidance expectations contributors project	1	0	lots of technical keywords like image, node, docker, guidance, expectations which could have led to misclassification
764	badge system	record achievements users contribution community	0	1	keywords like achievements and community could have led the model to be confused and misclassify
827	contribution	abstr_hyperlink abstr_hyperlink abstr_hyperlink abstr_hyperlink abstr_hyperlink ember javascript framework greatly reduces time	1	0	keywords like abstr_hyperlink, javascript, framework could be why it was misclassified
879	contributing	lots ways contribute people different expertise help various subprojects much apache bigtop deploy pacakging tools use puppet bootstrap set cluster recipes tools also welcome chef	1	0	unsure why it was misclassified despite having good indications in the content and heading that it is related to contribution
880	ctr model	bigtop supports commit review model development following rules used ctr process committer ahead commit patch without mandatory review felt confident quality reasonable testing done locally compilations pass rat check passed patch follows coding guidelines committer encouraged seek peer review advice hand doubts approach taken	1	0	has lots of content but no indication that it is for contribution because the content has words like commit, development, review, compilation which are all technical terms related to How category
1198	contributing	contribute framework please make sure checkout branch based development important allows accept pull request without publish public version release small	1	0	unsure why this is misclassified even though there are keywords like contributing and contribute
1404	contribute	contributions welcome report bugs issues find issue tracker grab source code pulii git repository	1	0	perhaps keyword like bugs and issues ended up being the reason for misclassification

Table 35. Contribution Category Misclassifications

ID	Heading	Content	Label	Pred	Reason
1569	contributing	submitting pull requests	1	0	not enough content for the model to make a sound decsision
1835	contributing	prs welcome fork	1	0	not enough content for the model to make a sound decsision
1880	contributing	guidelines	1	0	not enough content for the model to make a sound decsision
1886	contribute	kidsruby installer bootstrapped	1	0	not enough content for the model to make a sound decsision
1717	author	stefan walther abstr_hyperlink abstr_hyperlink abstr_hyperlink	1	0	there is author name and author keyword in the content and heading which could be why misclassification was done
2077	contributor code conduct	please note project released abstr_hyperlink participating project agree abide terms	1	0	keyword like released could have made the model choose a different category perhaps
2178	contributing	glad interested timessquare	1	0	not enough content for the model to make a sound decsision
2221	getting involved	open source project would love help prepared contributing guide help get started	1	0	content has keywords that could imply other categories like started, open source, project etc.

Table 36. Contribution Category Misclassifications

ID	Heading	Content	Label	Pred	Reason
3312	donation	donation abstr_hyperlink donations https github com simple rtmp server srs blob develop donations txt https github com simple rtmp server srs blob develop donations txt	1	0	has lots of technical keywords like github, com, server, develop which could be why it was misclassified
3588	donate	use love vegeta	1	0	not enough quality keywords to make a good decision
1520	include	include directives good take look theexamples directory get idea api feels ews cpp thin wrapper around microsoft ews api need refer abstr_hyperlink available parameters pass available attributes items abstr_number abstr_number looks like abstr_image items service use service whenever want talk exchange server please note one important caveat though ews cpp api designed blocking means whenever call one service member functions talk exchange server call blocks receives request	1	0	has lots of technical keywords like directives, include, parameters, hyperlink which could be why it was misclassified
3593	fullpage	abstr_image abstr_image abstr_hyperlink abstr_hyperlink abstr_image abstr_hyperlink abstr_number gziped simple easy use plugin create fullscreen scrolling websites also known single page websites onepage sites allows creation fullscreen scrolling websites	1	0	has lots of technical keywords like website, image, hyperlink which could be why it was misclassified
4163	license	abstr_hyperlink license freely received	1	0	the license keyword could have confused the model to classify it as a Who category

Table 37. Other Category Misclassifications

ID	Heading	Content	Label	Pred	Reason
4456	quotes	quotes blockquote first wiki engine consider worth using projects cite href dekorte blog blog cgi item amp steve dekorte cite blockquote blockquote looks like href atonie org git wiki git wiki may starting point need cite href tommorris org blog pid2761430 tom morris build perfect wiki cite blockquote blockquote makes git wiki cool backed git store clone wiki like could git repository always wanted wiki could pull offline access internets edit perhaps bulk favorite text editor git wiki allows cite href github willcodeforfoo git wiki wikis cloning wiki cite blockquote blockquote numerous people written diff merge systems wikis twiki even uses rcs used git instead repository would tiny could make personal copy entire wiki take plane sync changes back done cite href advogato org person apenwarr diary html git next unix cite blockquote	1	0	has lots of content with technical words like html, git, href, projects, internet which might be why it was misclassified

Table 38. Other Category Misclassifications

ID	Heading	Content	Label	Pred	Reason
4589	usage	first argument hav_tag accepts css xpath selectors supported hpricot body have_tag form action session body have_tag expectations placed upon inner text matched element providing another argument either string regexp body have_tag welcome body have_tag important blurb expectations placed upon number matched elements passing options hash body have_tag abbr count exactly one body have_tag minimum least body have_tag maximum body have_tag outgoing rspec count count key also accepts range making following equivalent body have_tag count body have_tag minimum maximum usage with_tag however longer supported instead block passed have_tag matched element successively yielded blocks return without raising expectationnotmeterror outer have_tag treated failed body have_tag thead thead thead have_tag count end also allows arbitrary expectations applied within block body have_tag sha1 inner_text length	1	0	its a lot of content that has ton of keywords like tag, css, selectors, body which could be why it was misclassified where those keywords are related to programming and/or development of a project

Table 39. Other Category Misclassifications

ID	Heading	Content	Label	Pred	Reason
4851	uses database sessions	bort setup use database store sessions default	1	0	quality of the content is not useful enough which could be why the model misclassified it
4876	liscencing	see copying	1	0	perhaps the licensing keyword could have led to the misclassification
4889	compile install	need compile install ruby configure exist older configure run autoconf generate configure run configure generate config makefile edit defines need usually step needed remove comment mark module names ext setup add module names present want link modules statically want compile non static extension modules probably architectures allow dynamic loading remove comment mark line option nodynamic ext setup run make optionally run make test check whether compiled ruby interpreter works well see message test succeeded ruby works hopefully run make install may super user install ruby fail compile ruby please send detailed error report error log machine type help others	1	0	there are lots of keywords like install, compile, ruby which could have led the model to classify it as a How category perhaps and another conclusion here could be that the manual labelling itself was misclassified by the labeler and this dataset does not fall under Other perhaps

Table 40. Other Category Misclassifications

ID	Heading	Content	Label	Pred	Reason
4922	gateway requirements	rule never want store credit card numbers means need gateway either provides automated recurring billing arb common offering credit card storage trustcommerce citadel freemium work gateway provides credit card storage preferred method work every gateway provides arb gateway provides arb mean application fire forget still needs know successful transactions send invoices failed transactions send warnings expire subscription order application know events without intervention arb module either needs send event notifications needs provide api retrieve review recent transactions freemium work arb modules provide api retrieve review recent transactions far safest route since gateways send email notifications must manually processed human ugh others unreliable event notification systems paypal see talklikeaduck denhaven2 articles cure paypal subscription blues case arb modules send event notifications hardly ever tell successful transactions still keep track periodic cycles send invoices makes whole arb thing barely useful really need list known good known bad gateways list beginning top head good gateways trustcommerce citadel use citadel arb braintree payment solutions securevault arb probably good gateways authorize net cim arb also offer transaction review api bad gateways loudcommerce linkpoint storage transaction review	1	0	this dataset has a lot of words in its content and there are plenty of keywords that could have led to the misclassification where the model does not know how to interpret this much data to make an accurate decision

Table 41. Other Category Misclassifications

ID	Heading	Content	Label	Pred	Reason
5004	requirements	universal feed parser feedparser org	1	0	the keyword requirements could have been a reason as to why it was not classified in Other category
5015	install	create suitable database import tables mysql lib table_setup sql use shell script lib nuke copy customise lib templates server_config template lib server_config php accordingly login admin change default admin password config administer users admin menu profit	1	0	there are lots of technical keywords in content like database, mysql, lib, script, config and more that could have made the model to choose What or How category as they are related to the development aspect of a project
5083	capistrano recipe	bort comes ready rock capistrano recipe setup based using git passenger ready multistage deployments deploys production config default need ignore update config deploy production deployment settings info capistrano ext multistage deployments found weblog jamisbuck org capistrano multistage	1	0	There are quite a few mentions of deployments in the content which could possibly be a reason as to why it was not classified as Other category

Table 42. Other Category Misclassifications

8.4 D - Confusion Matrices

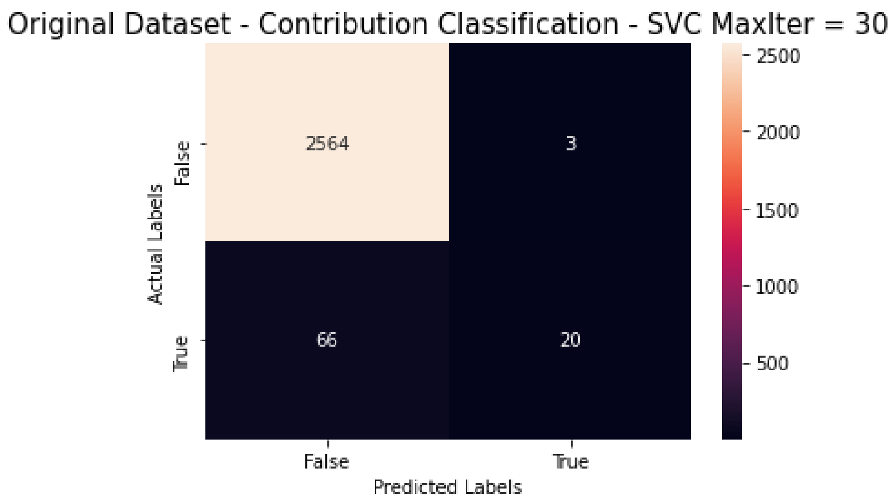


Fig. 14. Confusion Matrix - Original Dataset - Contribution Classification

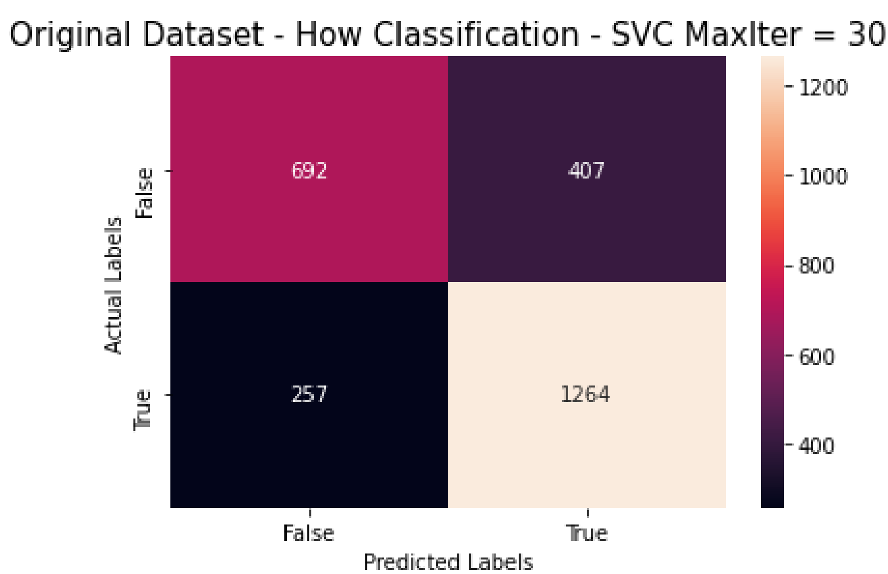


Fig. 15. Confusion Matrix - Original Dataset - How Classification

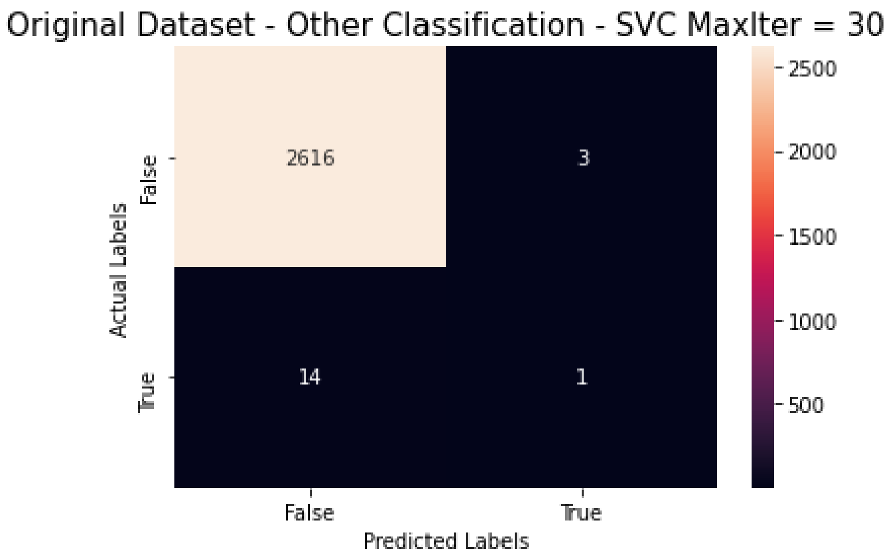


Fig. 16. Confusion Matrix - Original Dataset - Other Classification

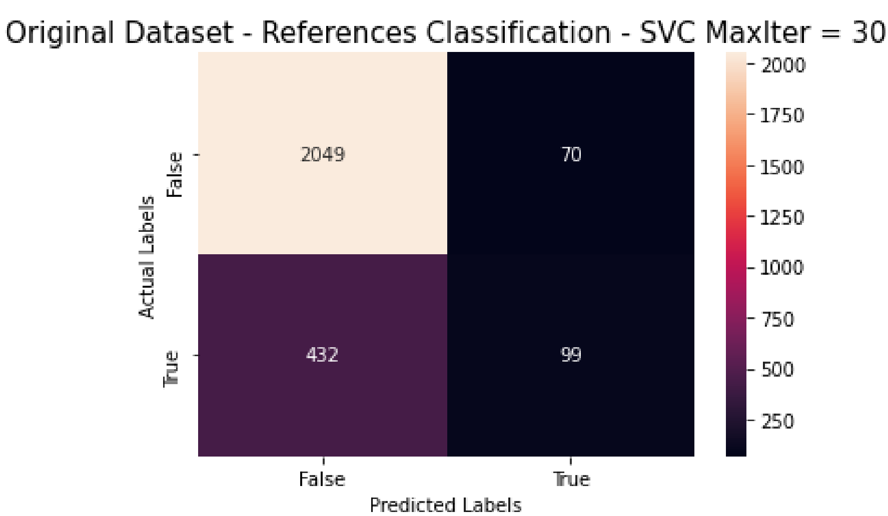


Fig. 17. Confusion Matrix - Original Dataset - References Classification

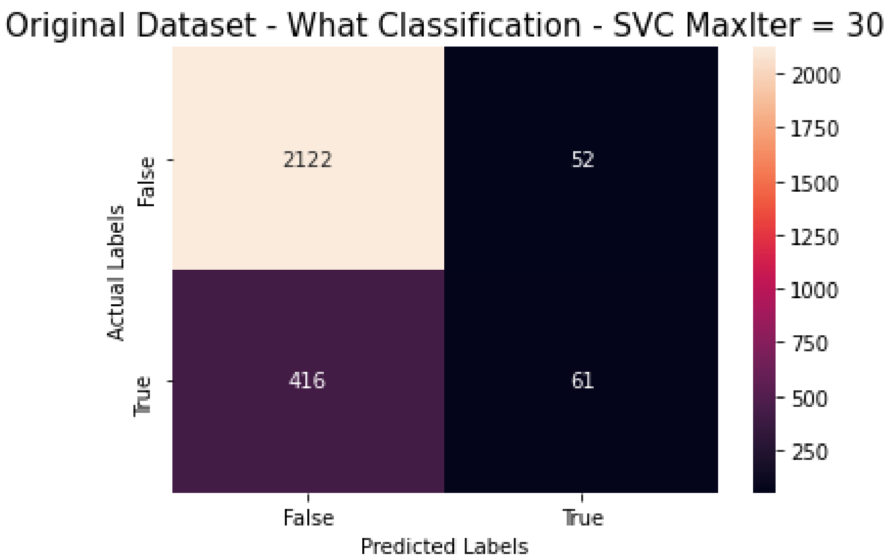


Fig. 18. Confusion Matrix - Original Dataset - What Classification

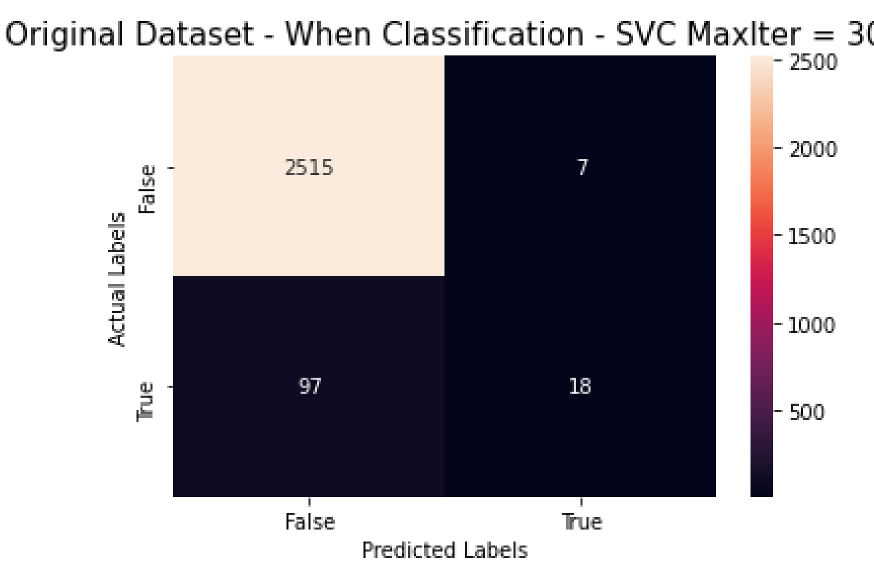


Fig. 19. Confusion Matrix - Original Dataset - When Classification

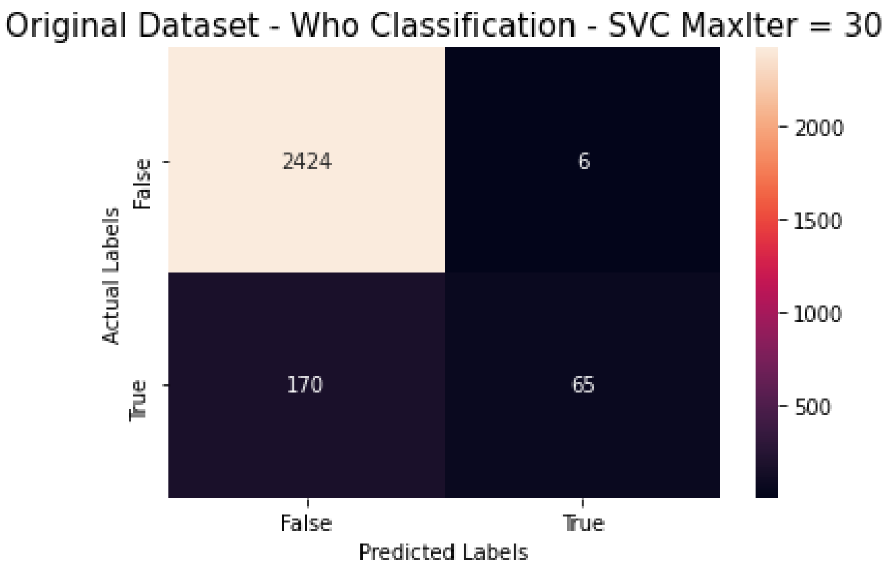


Fig. 20. Confusion Matrix - Original Dataset - Who Classification

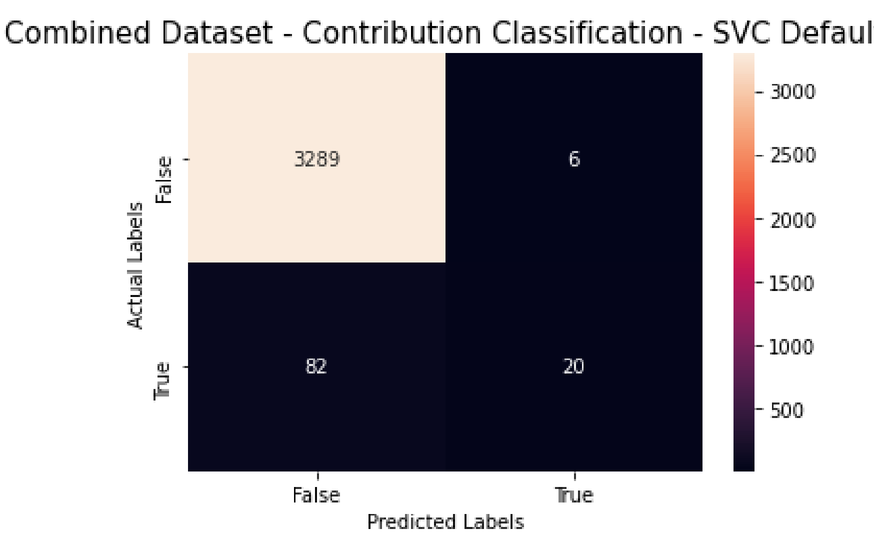


Fig. 21. Confusion Matrix - Combined Dataset - Contribution Classification

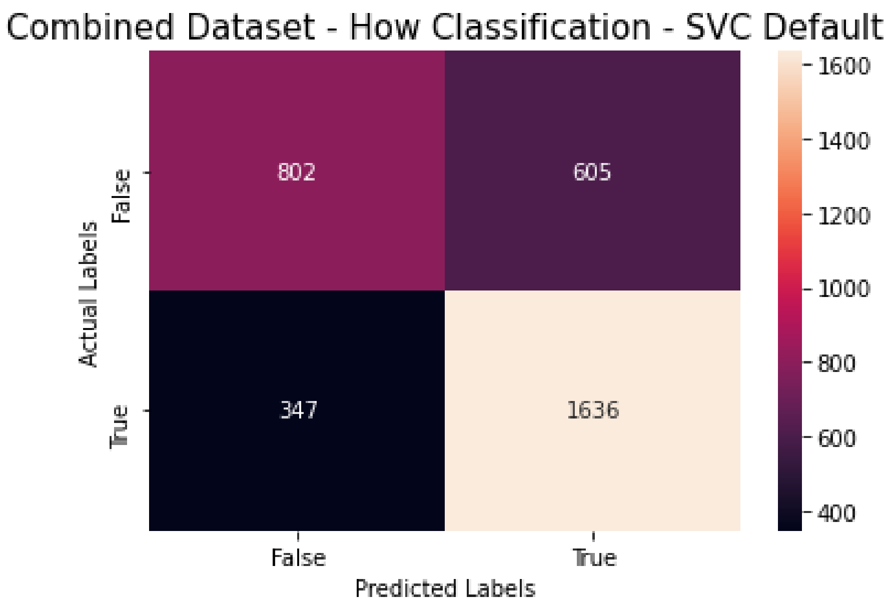


Fig. 22. Confusion Matrix - Combined Dataset - How Classification

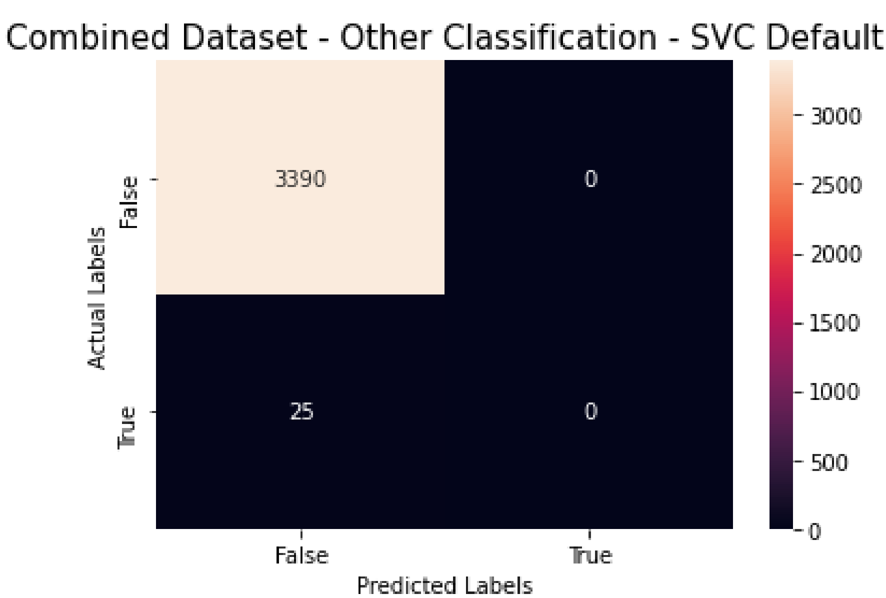


Fig. 23. Confusion Matrix - Combined Dataset - Other Classification

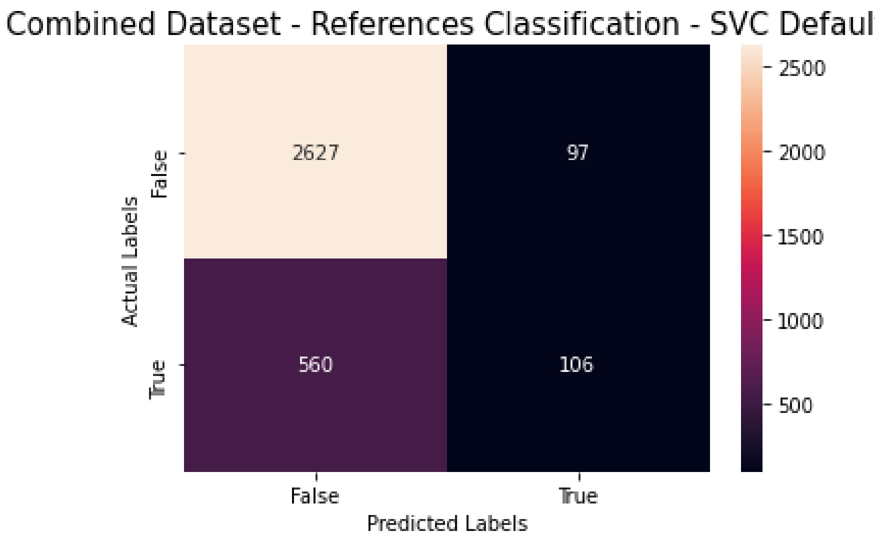


Fig. 24. Confusion Matrix - Combined Dataset - References Classification

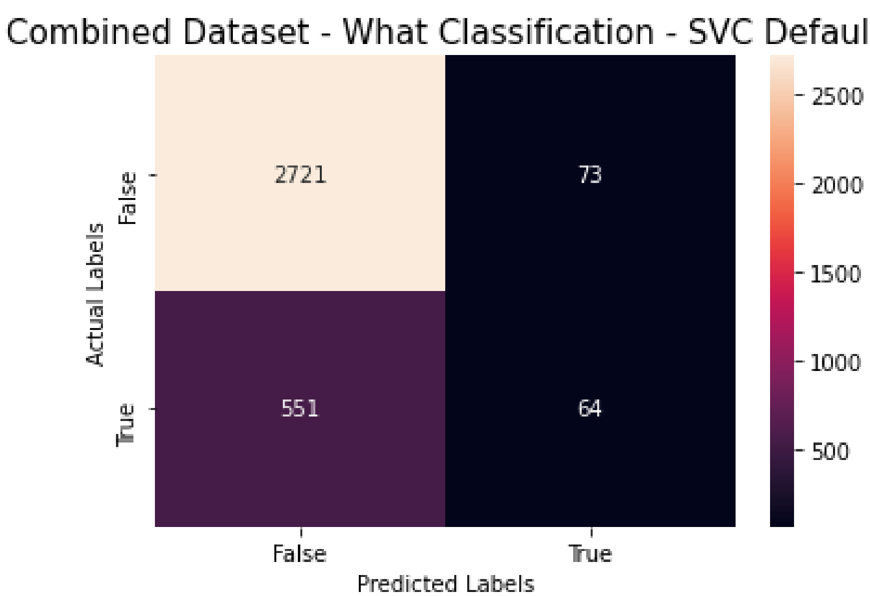


Fig. 25. Confusion Matrix - Combined Dataset - What Classification

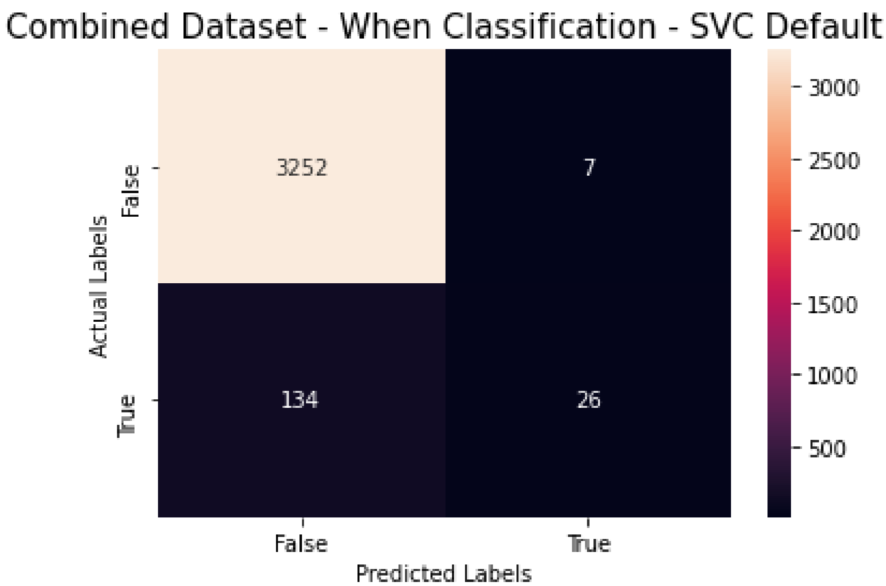


Fig. 26. Confusion Matrix - Combined Dataset - When Classification

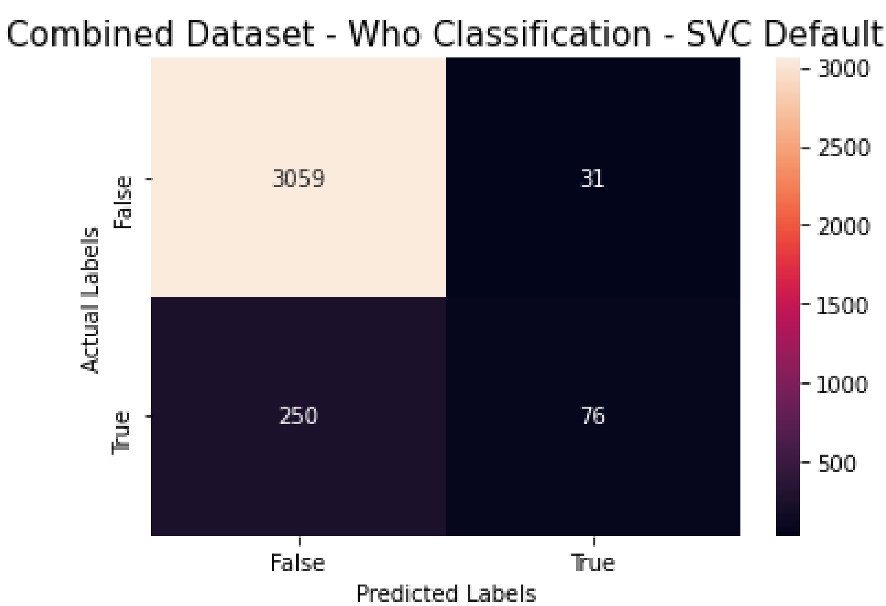


Fig. 27. Confusion Matrix - Combined Dataset - Who Classification

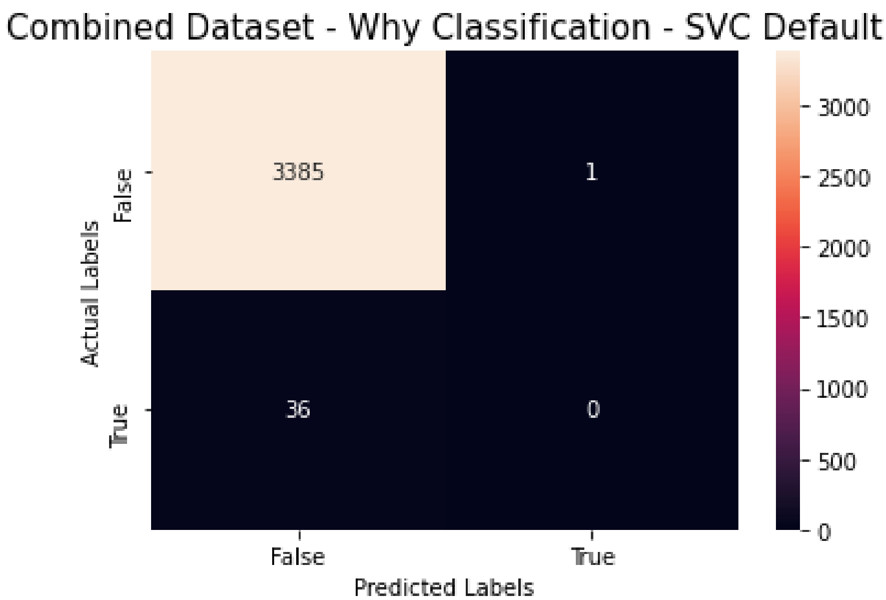


Fig. 28. Confusion Matrix - Combined Dataset - Why Classification