

2) INTRODUCTION

GitHub is a popular platform for hosting and collaborating on software development projects [1]. One key aspect of managing a project on GitHub is efficiently handling issues, which are reported by users to identify bugs, request features, or provide other feedback. In this paper, we propose a method for automating the classification of GitHub issues, which can help project maintainers prioritize and address them more effectively.

Manually classifying GitHub issues can be a time-consuming and error-prone process. Project maintainers must read each issue, understand its content, and assign it a label that accurately reflects its nature and importance. This task becomes increasingly difficult as the number of issues grows and the project becomes more complex. Automating the classification of GitHub issues can help project maintainers manage their workload more effectively and ensure that all issues are addressed in a timely and appropriate manner.

Our approach involves training a machine learning model on a dataset of labeled issues. The model is trained to predict the labels for novel issues as they are reported, based on the text of the issue and other relevant information. To create the dataset, we first collect 1000 issues from the React GitHub repository. We then manually label each issue with one or more categories, such as "bug", "feature", "support" or "other". Each issue is labelled by each of the group members and any issue that was found to have a tie was relabeled once again and discussed until consensus was found.

Once the dataset is created, the data is preprocessed by removing all headings, bolded text, and html comments since these constitute the React issue input template and do not reflect the content of the actual issue. Any URLs found had their final directory removed to leave only the general pathway. The base pathway was maintained since it was hypothesized that it would be relevant information for the model to know which sites the issues were linking to. Feature generation using TD-IDF was then introduced.

Once the dataset has been pre-processed and the new features generated, we used the training set (n=799 samples) to train a logistic regression model and Naïve Bayes machine learning model. We evaluated several different parameters using grid search and selected the parameters that performed the best on a held-out validation set (n=201). A full summary of how each hyperparameter from the grid search impacted the final results can be located in the Model Performance section below, but a clear best model was chosen based on the varying inputs. The trained model can now be used to automatically classify novel issues as they are reported on the project.

The total accuracy for the logistic regression classifier was 0.527 and F1 scores for each class are 0.583 for "Other", 0.544 for "Support", 0.194 for "Bug" and 0.200 for "Feature". This shows that our classifier performs the worst when attempting to classify issues that are feature related.

The total accuracy for the Naïve Bayes classifier was 0.507 and F1 scores for each class are 0.481 for "Other", 0.604 for "Support", 0.114 for "Bug" and 0.167 for "Feature". This shows that the Naïve Bayes performed worse than the logistic regression classifier, however, the difference is marginal.

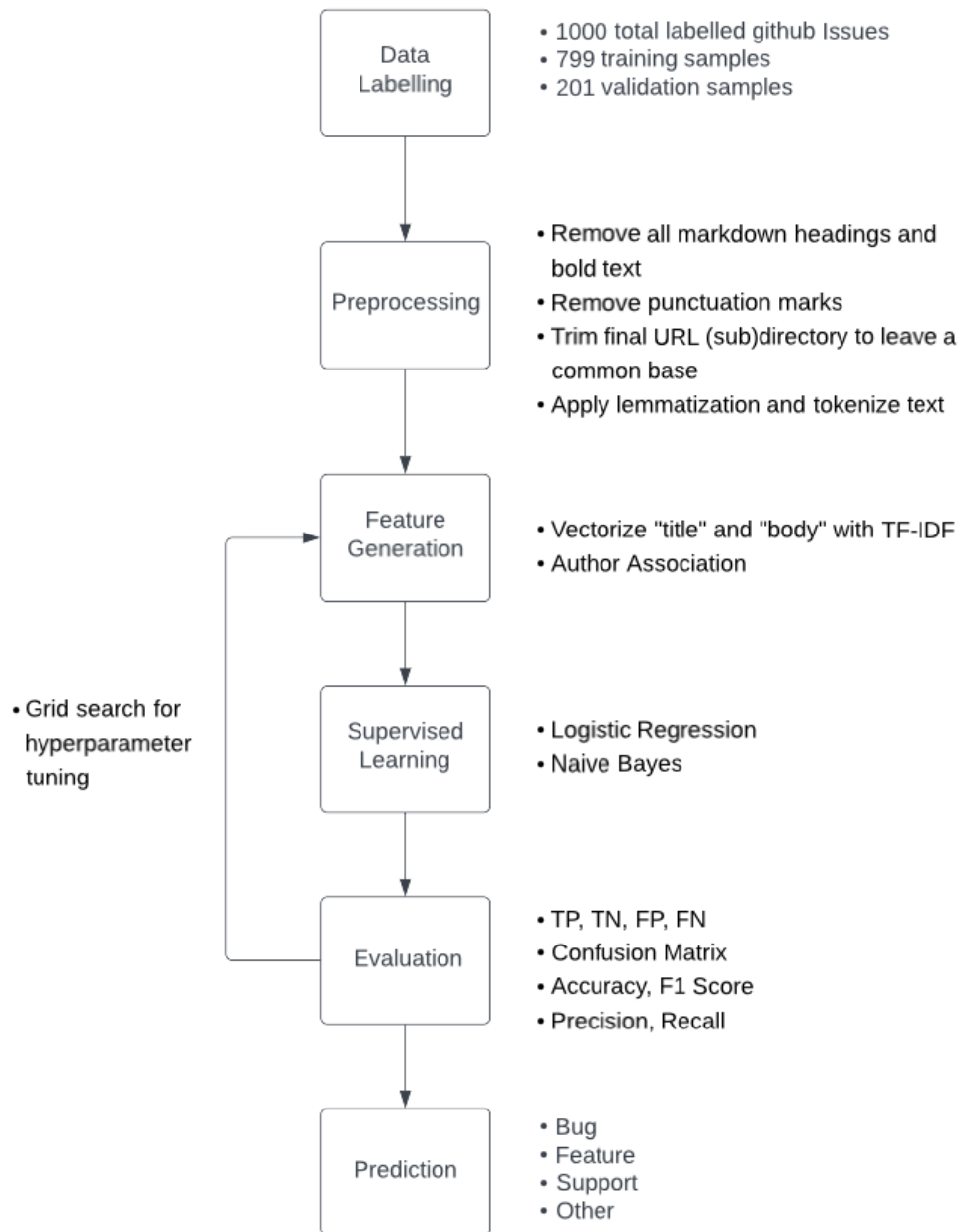


Figure 1: GitHub Issue Classification Machine Learning Pipeline

3) RESULTS – PRE-PROCESSING

a) Data collection and labelling

i) Approach

The data for this study was collected using a Python script that was developed to scrape all issues from any specified repository on GitHub using their API. A second Python script was then used to randomly select 1000 data samples from the collected data, with the samples evenly distributed across the last 5 years (2018 to 2022) of the React issues. This period was chosen instead of the 9-year history of the repository (2013-2022) to better represent the types of issues that may arise in the future since the types of issues posted in the infancy of the project were vastly different in structure and content.

Irrelevant columns were removed from the dataset, and the remaining columns used for initial data labeling are shown in the figure below.

```
# pick specific columns only (There are 111 in the original)
closed_issues = closed_issues[
    [
        "html_url",
        "number",
        "title",
        "labels",
        "state",
        "locked",
        "milestone",
        "comments",
        "created_at",
        "updated_at",
        "closed_at",
        "author_association",
        "state_reason",
        "assignee.login",
        "body",
    ]
]
```

Figure 2: Columns chosen to be labelled

The initial labels for the data in this study were chosen to be "bug" and "feature," with each category being further divided into "security" and "performance" subcategories. 50 labels were completed by each group member and applied to issues from the dataset's entire period. These labels were then discussed as a group to identify any discrepancies or inconsistencies. This process was done to identify any potential gaps in our assumptions about the data before the full dataset was labeled.

The results of the initial data labeling led to the transformation of our initial data labels to "bug," "feature," "support," and "other." However, due to inconsistencies in the labeling of the data, it was decided that

each group member would label all 1000 data samples, and any tiebreakers would be discussed as a group to reach a consensus on every one of the 1000 samples. This approach allowed us to ensure that the data was accurately and consistently labeled and provided a more reliable basis for our analysis.

ii) Results

The results of the initial labeling of the data in this study revealed a 5% discrepancy between classifications across the 1000 data samples, with the distribution of labels being shown in Figure 3. The discrepancy was discussed as a group, and a second and final pass of labeling was conducted to resolve any inconsistencies. Once the labeling was complete, the final distribution of labels was 4% "feature," 12% "bug," 40% "support," and 43% "other" as seen in Figure 4. This resulted in an unbalanced dataset, with the labels being skewed towards the "support" and "other" categories.

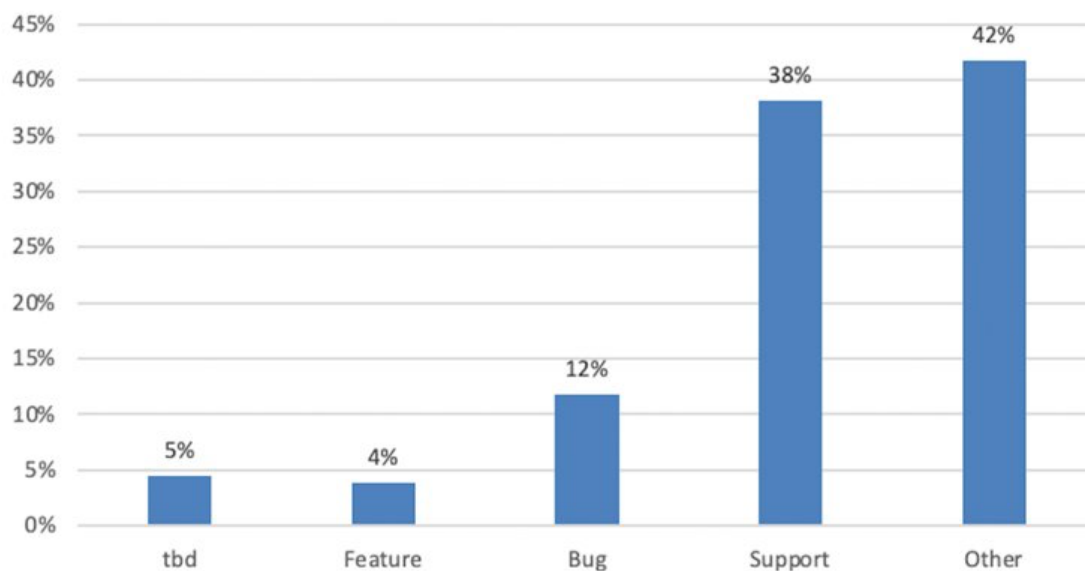


Figure 3: First pass distribution of labels across 1000 data samples

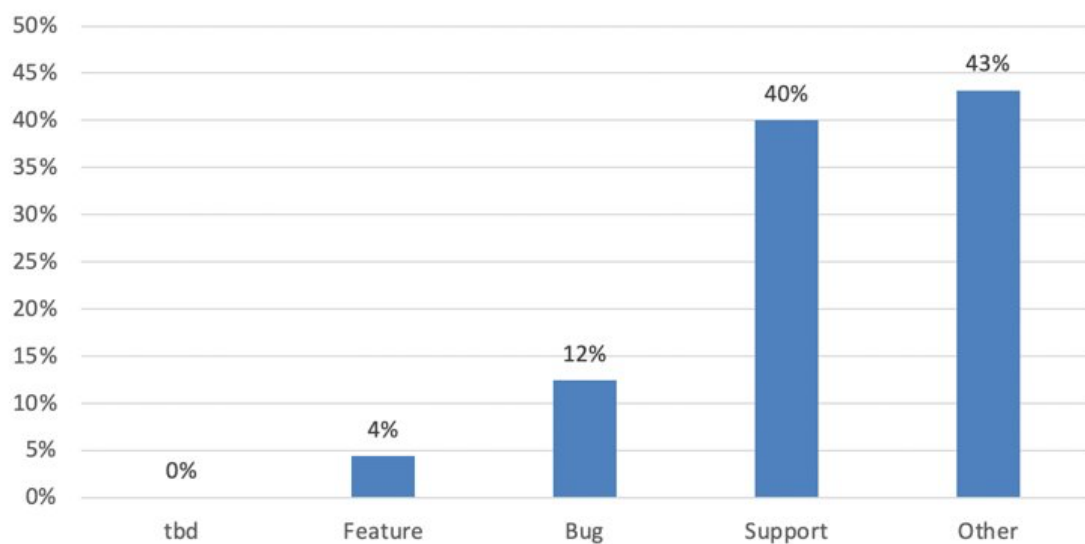


Figure 4: Final label distribution across 1000 data samples

In Figure 5, an example of a labeling disagreement can be observed. The user jacmkno raised the issue titled "No good reasons to remove string refs from react-native" and the team questioned whether this issue should be labeled as "support" or "other". After discussion, the team proposed that since the issue was closed without comment from support staff, it should be labeled as "other" instead of "support".

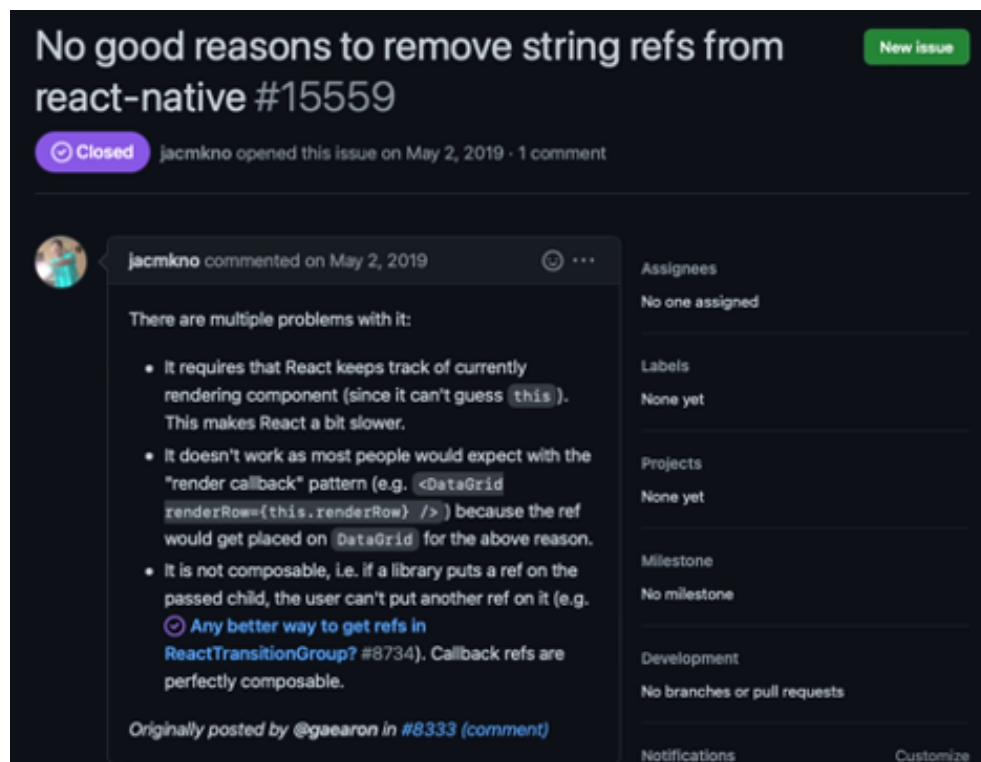


Figure 5: Issue #15559 from user jacmkno

In figure 6, another example of a labeling disagreement can be observed. The support staff member user gaaaron raised an issue titled "Add more Fuzz Tests" which asked for support on a new addition to the testing suite and the team questioned whether this should be regarded as a "support" or "feature". After discussion, the team proposed that since this issue was raised by the support staff to ask for assistance where the outcome would be a new feature to the codebase it should be classified as "feature" and not "support".

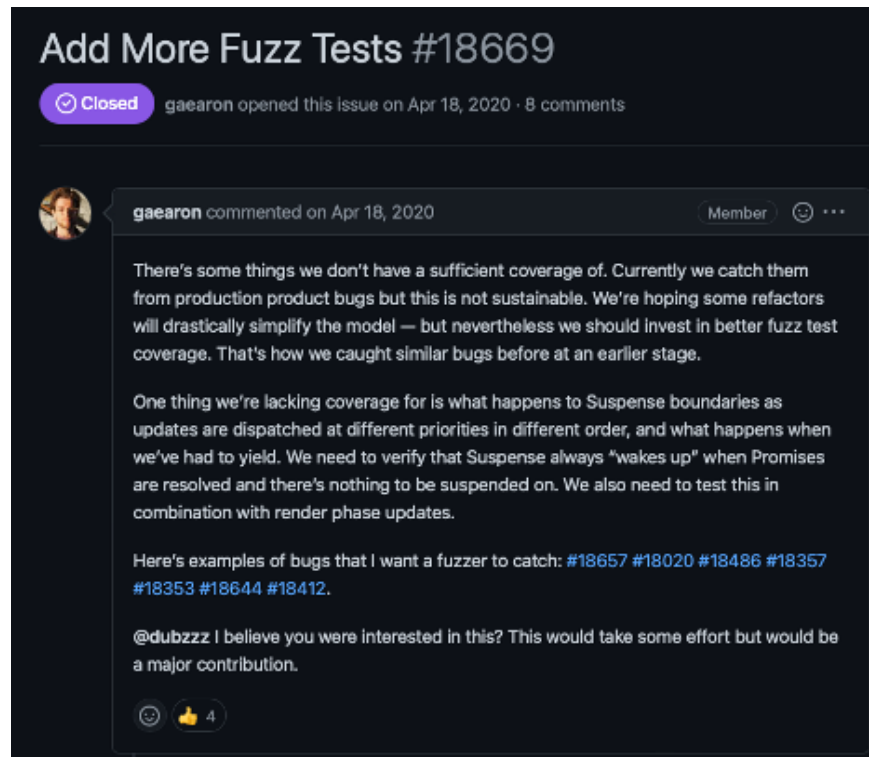


Figure 6: Issue #18669 from user gaearon

In figure 7, another example of a labeling disagreement can be observed. The user *sophiebits* raised an issue titled “DevTools: Uncaught error doesn’t go away on page refresh” which asked for support on an error in the React devtools and the team questioned whether this should be regarded as a “support” or “bug”. After discussion, the team proposed that since this issue was discussed by the support staff in the comments and a pull request was merged that referenced this issue that this should be labelled as “bug” and not “support” as this was a bug in the React code and not an issue with the user’s code.

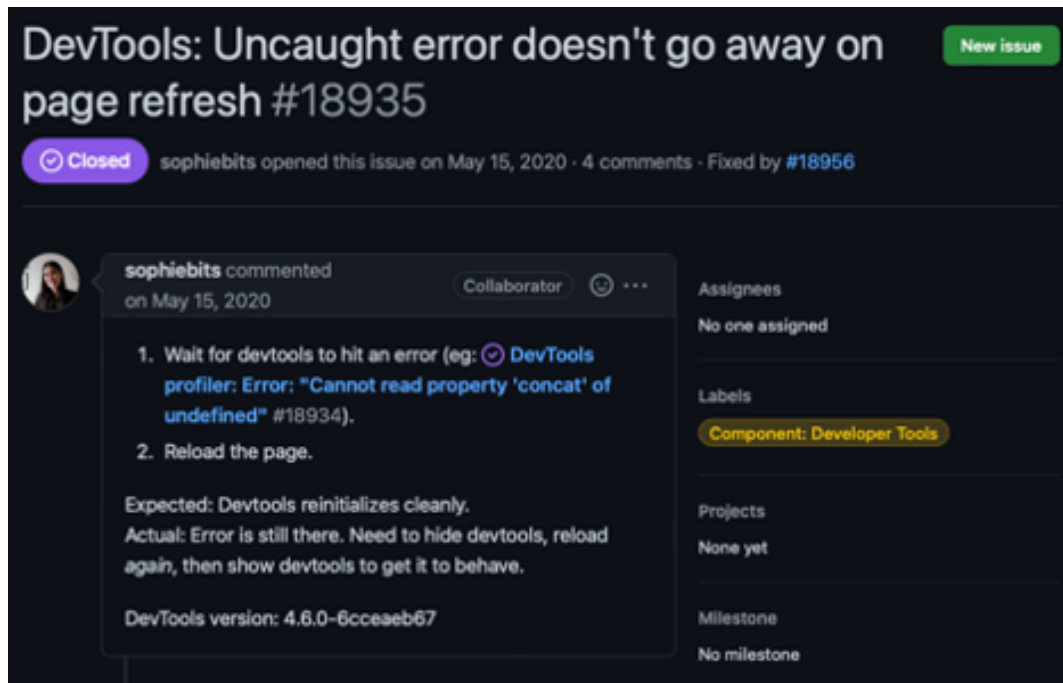


Figure 7: Issue #18935 from user sophiebits

b) DATA PREPROCESSING

i) Approach

Pre-processing text involves a series of steps used to clean and prepare it for further analysis. This is a key step in many natural language processing tasks, as it helps to simplify the text and make it easier to analyze. Pre-processing can involve various steps, including removing formatting, removing punctuation, applying lemmatization and vectorizing using TF-IDF [2].

The first step in pre-processing GitHub issue samples is the removal of markdown headings and bold text from the title and body. These elements are often part of the issue template from the React team, rather than the actual issue report, and removing them can help focus on the issue's content. This can be particularly useful when analyzing customer feedback, as it allows the analysis to focus on the specific issues that customers are reporting, rather than the formatting of the text.

The second step is the removal of punctuation marks. This can help to simplify the text by removing unnecessary characters, such as commas, periods, and exclamation points [2]. This can make it easier to analyze the text, as it reduces the amount of "noise" that can distract from the meaning of the words.

Once the punctuation marks have been removed, the text can be trimmed to a common base. This involves removing the final URL (sub)directory, which can help to standardize the text and make it easier to compare. For example, if the text includes multiple URLs, trimming them to a common base can help to ensure that they are all referring to the same location, making it easier to analyze the context in which the URLs are used. An example of this is that a number of issues would link to issues on the GitHub repository for Create React App (<https://github.com/facebook/create-react-app>), so we decided to keep

the URL to the repository as a feature, but we removed the part of the link that lead to a specific post, as that would be too specific.

The next step in pre-processing the text is to apply tokenization and lemmatization. This is a technique used to tokenize the text, which involves converting the words to their base forms [2]. This can make it easier to identify and analyze the key components of the text, as it allows similar words to be grouped together. For example, "running" and "ran" would both be converted to the base form "run," making it easier to analyze the context in which the word is used.

Based on the lemmas produced in the previous stage, unigrams (each single word) and bigrams (all pairs of two words) were generated to add additional features to the final feature vector. While it's likely that unigrams on their own may be the most impactful, bigrams were added to potentially add further nuance to the text.

The final step in pre-processing is to apply term frequency-inverse document frequency (TF-IDF) to the "body" and "title" columns. The use of a TF-IDF Vectorizer is a common technique in natural language processing tasks, as it allows text to be represented as numerical feature vectors [2]. TF-IDF assigns a "weight" to each ngram that reflects its relative importance to the document (specific issue) in the corpus (all issues). The advantage of this technique over a simple token counter is that it allows less important words (like stop words) to naturally be ignored, while placing greater focus on the terms that should hold the most importance to the model. In this case, the vectorizer will be used to convert the Title and Body of each GitHub issue into a feature set, which can then be used as input to machine learning algorithms. One advantage of using a vectorizer is that it allows the Title and Body of each issue to be treated as separate features, rather than a single, concatenated string. This can be useful, as the Title and Body may have distinct levels of importance or impact on the outcome of the analysis. For example, the Title may contain more concise and relevant information about the issue, while the Body may provide additional context or details. By treating the Title and Body as separate features, the model can more accurately capture and analyze the information contained in each.

Finally, one additional feature was added on top of the text data. The "author_association" for each issue was added to the feature vector with all the results from the TF-IDF pre-processing. In our dataset, issues were posted by four distinct categories of authors: Members, Collaborators, Contributors, and None. It was hypothesized that certain types of members in the React community may post more valid bugs or feature requests than others, so we included them as features for the classifier. A one-hot-encoder was used to process these different categorical values into numerical values for the classifiers to use in training and predicting [2].

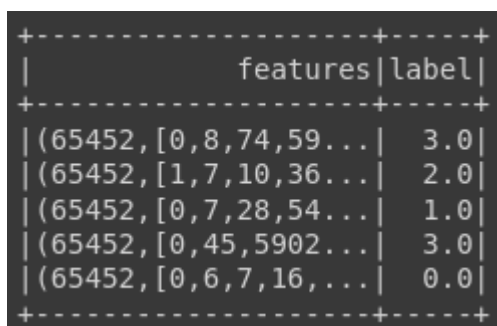
Overall, pre-processing text is an important step in many natural language processing tasks. It helps to simplify and clean the text, making it easier to understand and interpret. By removing unnecessary formatting, punctuation, and trimming the text to a common base, pre-processing can make it easier to focus on the key components of the text and analyze them more effectively.

ii) Results

The input to the pre-processing stage was a CSV file with 15 columns (see Figure 2 above) and 1000 samples. The output from the pre-processing stage was a dataframe with the same 1000 samples, but just two columns, a features vector and a label column which is ready to be consumed by a PySpark ML classifier.

The label column is simply the 'Target' column from our input, transformed by the pysparkml StringIndexer into corresponding integers for each label.

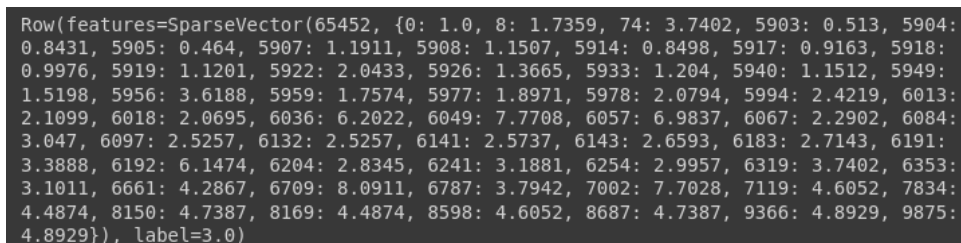
The features vector is the result of all the pre-processing steps in the section above. Each sample contains a SparseVector with 65452 elements, where each element corresponds to a different generated feature. The first 5 rows of the resultant dataframe can be seen in the figure below:



features	label
(65452, [0, 8, 74, 59...]	3.0
(65452, [1, 7, 10, 36...]	2.0
(65452, [0, 7, 28, 54...]	1.0
(65452, [0, 45, 5902...]	3.0
(65452, [0, 6, 7, 16, ...]	0.0

Figure 8: Final pre-processed dataframe

The full first row, with the full SparseVector feature can be seen in the figure below. It should be noted that each entry in the features column does not contain all 65452 elements, but only the non-zero elements in order to save space.



```
Row(features=SparseVector(65452, {0: 1.0, 8: 1.7359, 74: 3.7402, 5903: 0.513, 5904: 0.8431, 5905: 0.464, 5907: 1.1911, 5908: 1.1507, 5914: 0.8498, 5917: 0.9163, 5918: 0.9976, 5919: 1.1201, 5922: 2.0433, 5926: 1.3665, 5933: 1.204, 5940: 1.1512, 5949: 1.5198, 5956: 3.6188, 5959: 1.7574, 5977: 1.8971, 5978: 2.0794, 5994: 2.4219, 6013: 2.1099, 6018: 2.0695, 6036: 6.2022, 6049: 7.7708, 6057: 6.9837, 6067: 2.2902, 6084: 3.047, 6097: 2.5257, 6132: 2.5257, 6141: 2.5737, 6143: 2.6593, 6183: 2.7143, 6191: 3.3888, 6192: 6.1474, 6204: 2.8345, 6241: 3.1881, 6254: 2.9957, 6319: 3.7402, 6353: 3.1011, 6661: 4.2867, 6709: 8.0911, 6787: 3.7942, 7002: 7.7028, 7119: 4.6052, 7834: 4.4874, 8150: 4.7387, 8169: 4.4874, 8598: 4.6052, 8687: 4.7387, 9366: 4.8929, 9875: 4.8929}), label=3.0)
```

Figure 9: Single sample from the pre-processed dataframe

The words with the highest frequencies in the issue titles and bodies have been shown in the figures below. This illustrates, as expected, that most of the highest frequency terms are stop words (and other very common words related to the React project). These words will all naturally get lower weightings from the TF-IDF process.

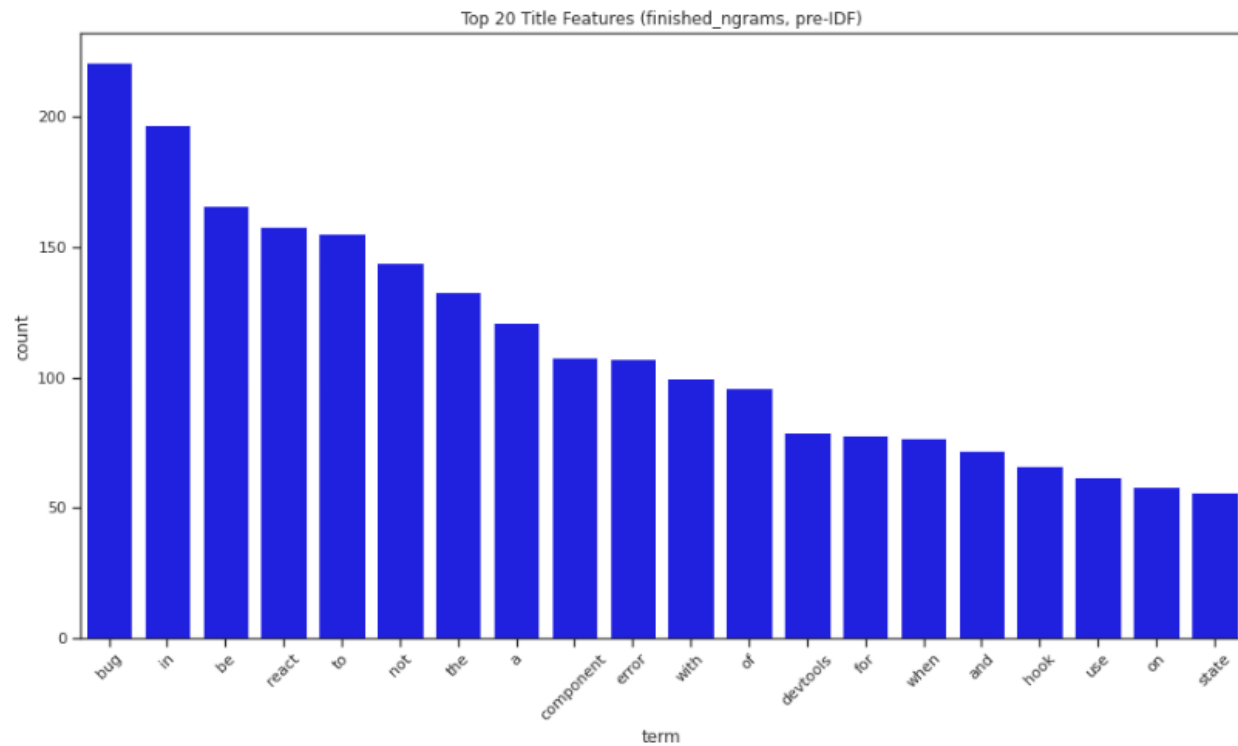


Figure 10: Top 20 Title Features

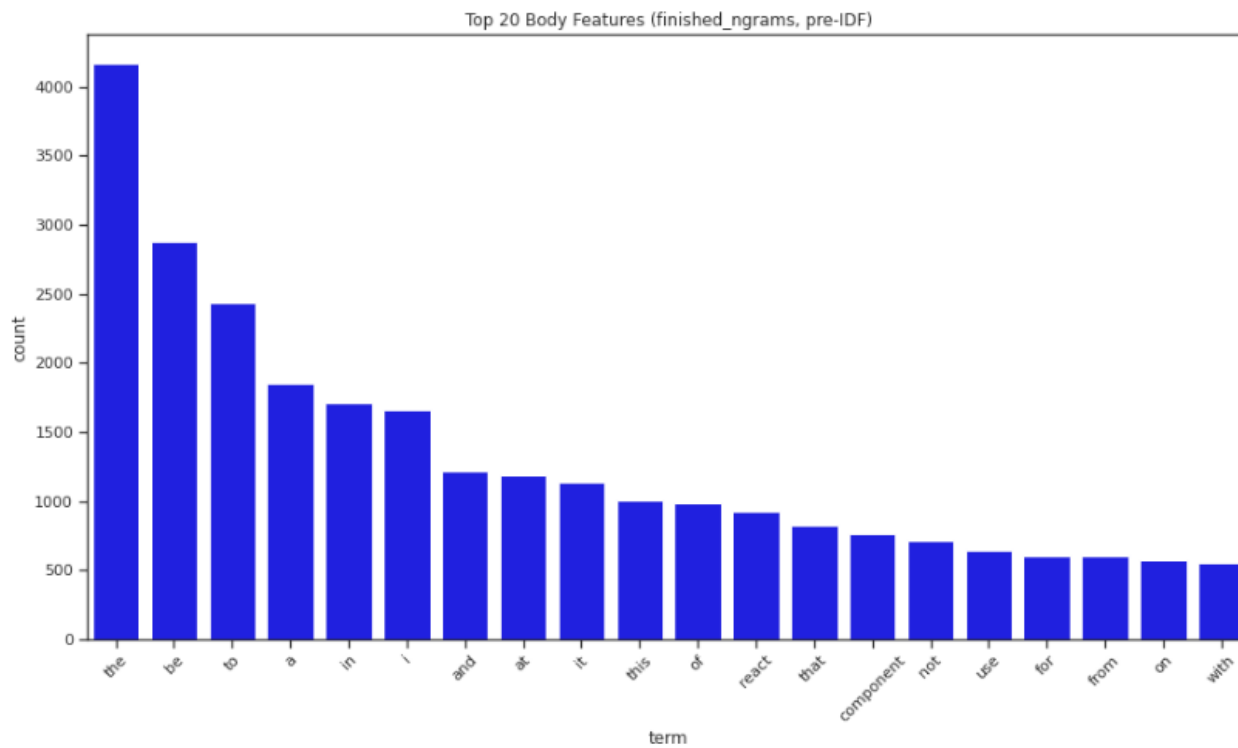


Figure 11: Top 20 Body Features

3) RESULTS - ISSUE CLASSIFICATION

Two (2) supervised learning classification algorithms were employed on this project:

- Logistic Regression
- Naïve Bayes

These models were primarily chosen because they can perform well on large high-dimensional sparse datasets, can be applied to multiclass problems, and can be trained and employed to obtain predictions quickly. Further, linear models such as the Logistic Regression classifier tend to perform well when the number of samples is small relative to the feature set [2]. Also worth noting, the Naïve Bayes family of classifiers, while based on simple assumptions, have been considered effective and powerful classifiers [3], and have performed well in complicated real-world scenarios [4].

The ensuing sections will present results and commentary associated with the application of the selected classifiers. Our focus will be on the Logistic Regression classifier, followed by the Naïve Bayes classifier serving as a baseline/reference – it is expected that the former will outperform the latter.

Per convention, modeling and evaluation work carried out on the project followed the process depicted below. After pre-processing, data was split 80/20 - the 20% test set was held out for final evaluation, which followed hyperparameter selection via grid search cross validation, and model retraining.

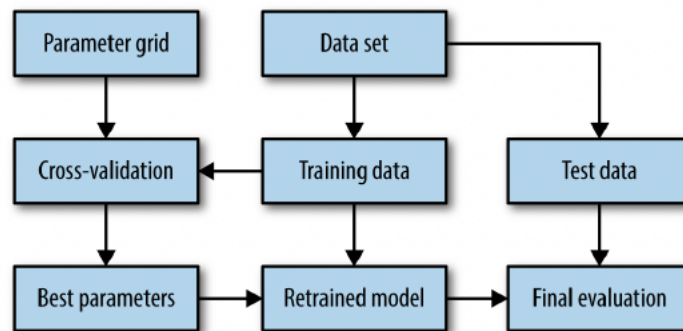


Figure 12: Hyperparameter Selection and Model Evaluation Overview [2]

d) 1) Model Performance

i. / ii. Approach & Results -- Logistic Regression Classifier

Initial Model Fitting and Prediction: Training Data – default parameters (Case 1)

The model was fit using the training data, and then the fitted-model was used to make label predictions – in this case, the model predicted with 100% accuracy (i.e., overfitting) as shown below in the figure and summary table.

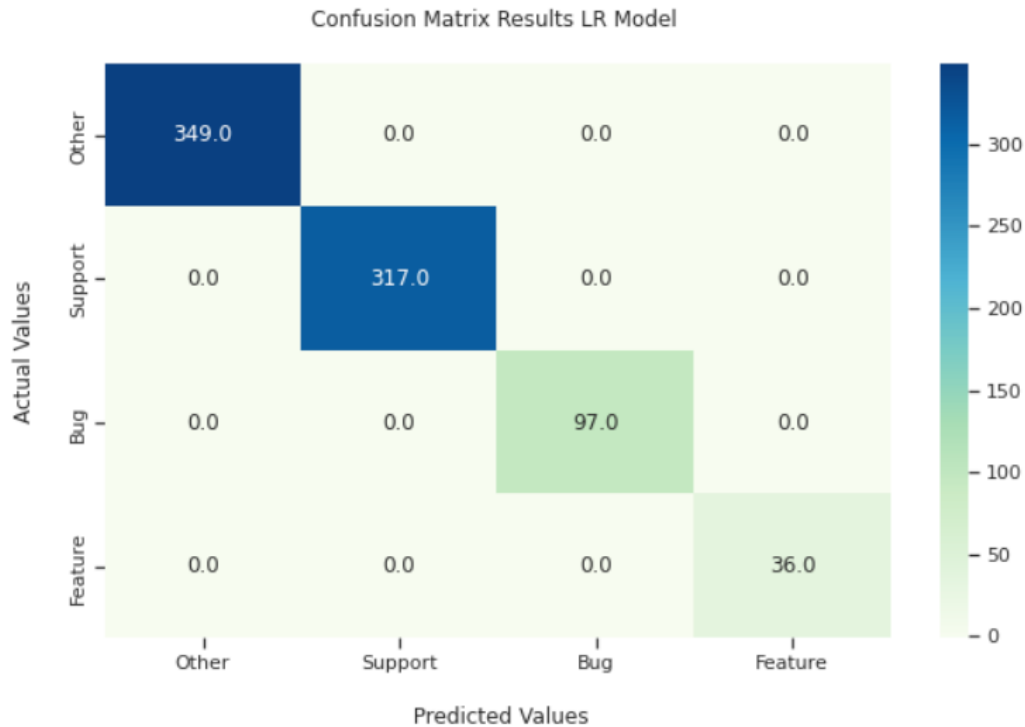


Figure 13: Confusion Matrix – Initial Model Predictions w/ Training Data (over-fitting) (**Case 1**)

Table 1: Initial Model Predictions w/ Training Data (**Case 1**)

	Variable	Value
Dataset	Training	-
Model	LogisticRegression()	-
Params	regParam	0.0
	elasticNetParam	0.0
Metrics	Accuracy	1.00
	F1 Score	1.00

Overfitting in this case is not unexpected, since linear models without an imposed regularization penalty (regParam=0.0 is default) are known to overfit high-dimensional data. Regularization parameters are key to ensuring a linear model does not overfit, and as a result we will focus on these parameters in our grid search cross validation work.

Initial Model Fitting and Prediction: Test Data – default parameters (Case 2)

As can be observed from the figure below, the model is not generalizing well (i.e., high variance), and label prediction accuracy is relatively low. Of note, as with the training dataset (due to a stratified data splitting), the bulk of the data resides in the “Other” and “Support” labels (approximately 83%). Also,

misclassifications appear to be worse for “Bug” and “Support” labels in this scenario, with ~63% and ~59% misclassifications, respectively. Overall scoring metrics are included in the table below.

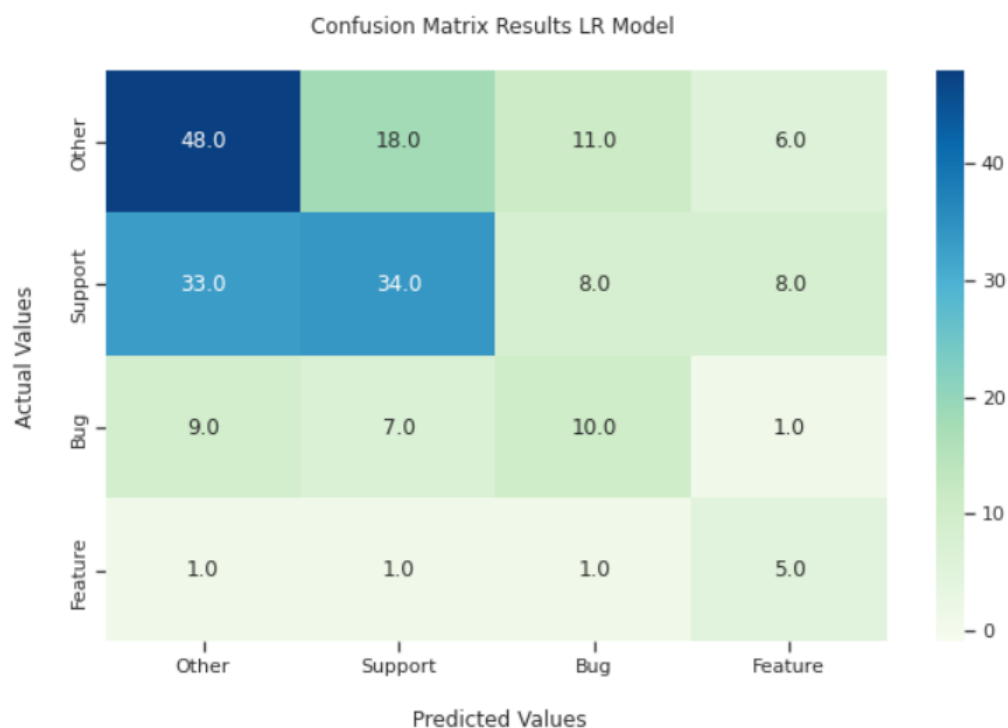


Figure 14: Confusion Matrix – Initial Model Predictions w/ Test Data (**Case 2**)

Based on the results from the confusion matrix above, the False Negative (FN), False Positive (FP), True Negative (TN), and True Positive (TP) have been shown for each class in the table below. This considered the perspective from each class individually, taking each class as their own positive class, and all other classes as the negative class.

Table 2: Confusion Table – Initial Model Predictions w/ Test Data (**Case 2**)

	Other	Support	Bug	Feature
FN	27	37	24	7
FP	53	40	1	1
TN	65	78	173	192
TP	56	46	3	1

Table 3: Initial Model Predictions w/ Test Data (Case 2)

	Variable	Value
Dataset	Test	-
Model	LogisticRegression()	-
Params	regParam	0.0
	elasticNetParam	0.0
Metrics	Accuracy	0.483
	F1 Score	0.486

Logistic Regression Model Cross Validation via Grid Search

Based on the results of our initial model fitting and predictions with the training and test data, it was evident that hyperparameters needed to be tuned. Based on over-fitting, we focused on two (2) regularization parameters: "regParam" and "elasticNetParam". A table is provided below summarizing the effects and relation between these two (2) parameters; further details can be found on GitHub [5].

Table 4: Regularization Parameters - Penalty Summary Table

regParam (λ)	elasticNetParam (α)	Regularization Penalty
$\lambda = 0$	$\alpha = 0$	None
$\lambda > 0$	$\alpha = 0$	Ridge (L2)
$\lambda > 0$	$\alpha = 1$	Lasso (L1)
$\lambda > 0$	$0 < \alpha < 1$	Blended penalty

To find the best set of parameters a grid search approach was taken using PySparkML (numFolds=5). From the table and heatmap figure below, we can see the best parameter set (regParam=0.1, elasticNetParam=0.0) – i.e., L2 penalty.

Table 5: Grid Search Results (average f1-Scores)

elasticNetParam	0.0	1.0
regParam		
0.0	0.482941	0.482941
0.01	0.491709	0.489146
0.1	0.497888	0.264843
1.0	0.488745	0.264843
10.0	0.490779	0.264843

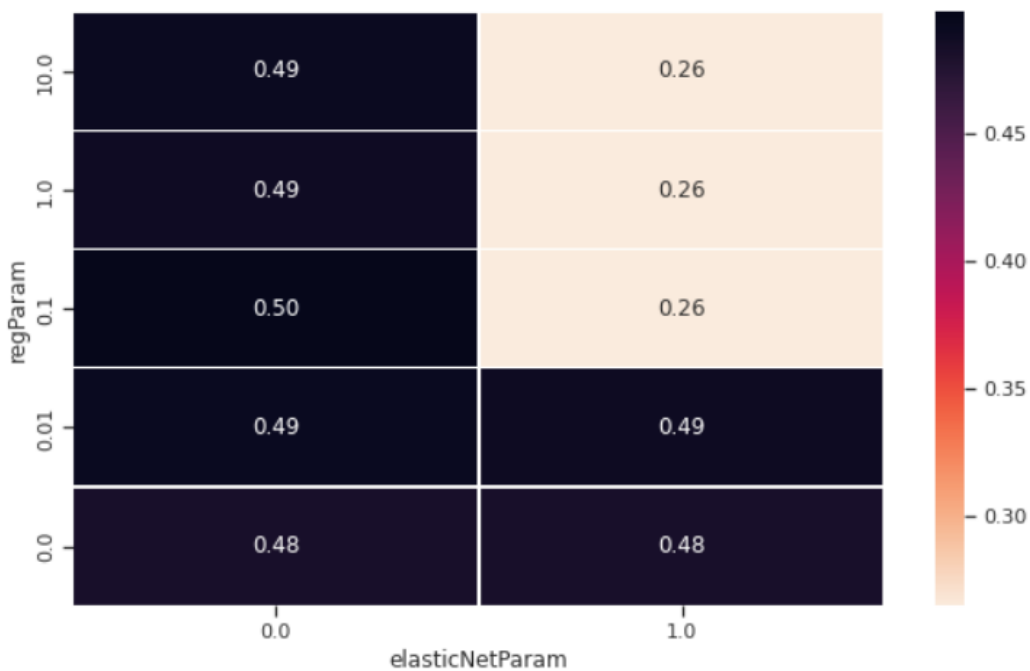


Figure 15: Heatmap – Logistic Regression Grid Search Results (average f1 scores)

e) 1) Model Performance Based on Hyperparameters Selection

i. / ii. Approach & Results -- Logistic Regression Classifier

Model Fitting and Prediction: Test Data – best parameters (Case 3)

After finding the best hyperparameters of interest, the best model (which is also found via the grid search cross validation) was used again to make predictions on the test set. Refer to confusion matrix figure and case summary table below.

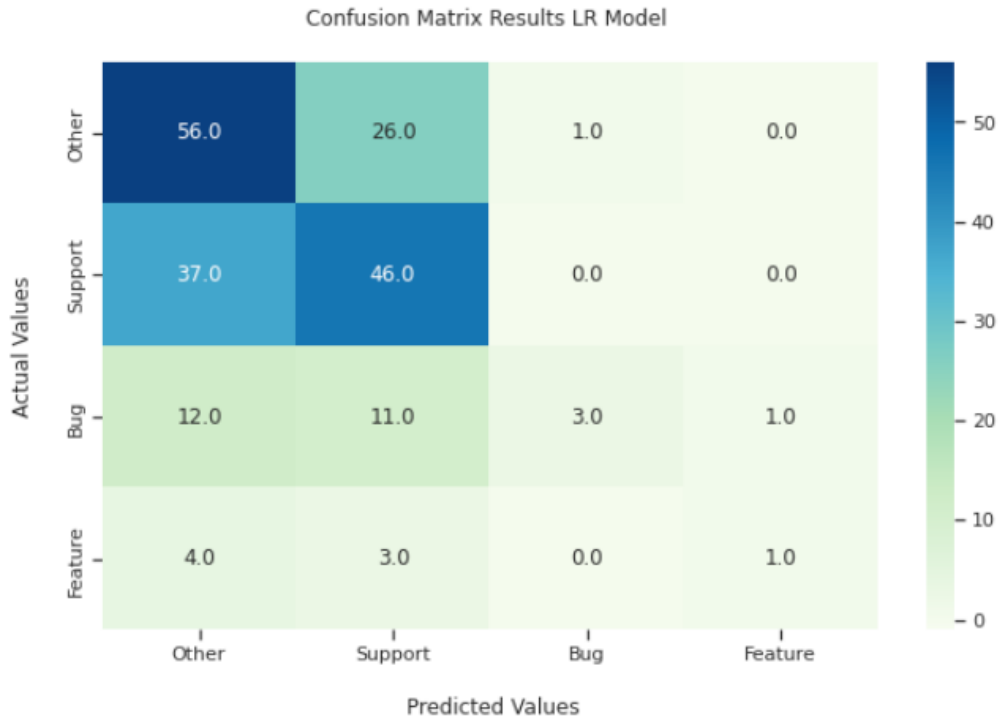


Figure 16: Confusion Matrix – Best Model Predictions w/ Test Data (**Case 3**)

Based on the results from the confusion matrix above, the False Negative (FN), False Positive (FP), True Negative (TN), and True Positive (TP) have been shown for each class in the table below. This considered the perspective from each class individually, taking each class as their own positive class, and all other classes as the negative class.

Table 6: Confusion Table – Best Model Predictions w/ Test Data (**Case 3**)

	Other	Support	Bug	Feature
FN	27	37	24	7
FP	53	40	1	1
TN	65	78	173	192
TP	56	46	3	1

Following cross validation, an overall improvement in model accuracy/score is observed, as expected. However, the improvement is relatively small, and in terms of specific label predictive capability, the model has improved with respect to “Other” and “Support” labels but decreased for “Bug” and “Feature” labels.

Table 7: Best Model Predictions w/ Test Data (Case 3)

	Variable	Value
Dataset	Test	-
Model	LogisticRegression()	-
Params	regParam	0.1
	elasticNetParam	0.0
Metrics	Accuracy	0.527
	F1 Score	0.500

Logistic Regression Classifier Metrics (best parameters - Case 3)

Performance metrics for the best parameter Logistic Regression model are summarized in this section. Individual metric-based conditional formatting has been used in the summary table below to highlight the performing vs. non-performing areas.

Overall Accuracy (total correct label predictions / total target labels) = **0.527**

Table 8: Best Model w/ Test Data: Performance Metrics (Case 3)

	Other	Support	Bug	Feature
Recall	0.674	0.554	0.111	0.125
Precision	0.514	0.535	0.750	0.500
f1 Score	0.583	0.544	0.194	0.200

Based on the small data set and imbalances present, the model can be considered to be providing reasonable performance for “Other” and “Support” labels, while generally performing poorly with respect to “Bug” and “Feature” labels.

The precision metric is a measure of the positive predictions. The higher the precision value is, the lower the count of the false positives. Looking at the results of the model’s predictions, we can see that the precision is above 50% for all classes, so this means that for all the predictions made for each class, at least 50% of them were correct. The ‘Bug’ label had the highest precision value, so it is the least likely to get false positive predictions.

The recall metric is a measure of the negative predictions. The higher the recall value, the lower the count of the false negatives. The ‘Other’ and ‘Support’ labels had pretty good recall values, meaning that there is a lower chance of getting a false negative when predicting these labels. The ‘Bug’ and ‘Feature’ labels have very poor recall, so there is a high chance that when predicting an issue that truly should be a ‘Bug’ or ‘Feature’, it will be predicted as a different label altogether.

The F1 metric is a measure that combines both precision and recall, balancing the two metrics. In our application, neither false negatives nor false positives are critical to the outcome, so the F1 score might be the best score to consider for overall performance. The poor recall scores for the ‘Bug’ and ‘Feature’ labels are evident in their F1 scores, and the ‘Other’ and ‘Support’ labels perform much better overall.

Logistic Regression Classifier Case Summary

Logistic Regression model cases are summarized in the table below for convenience and comparison purposes.

Table 9: Logistic Regression Model Case Summaries

		Case 1	Case 2	Case 3 (best params)
	Variables	Values	Values	Values
Dataset	Train / Test	Train	Test	Test
Params	regParam (λ)	0.0	0.0	0.1
	elasticNetParam (α)	0.0	0.0	0.0
Metrics	Accuracy	1.00	0.483	0.527
	F1 Score	1.00	0.486	0.500

The comparison clearly shows an improvement in model predictive performance following retraining with the best parameters found via grid search cross validation. However, the model still has high bias, and further work and improvements would be required before putting it into service - for example, collection and training on a larger and possibly more balanced dataset, in concert with exploration of a larger set of hyperparameters.

d) 2) Model Performance

i. / ii. Approach & Results -- Naïve Bayes classifiers

The family of Naïve Bayes (NB) supervised learning algorithms are based on Bayes' theorem with the simplifying assumption of feature pair conditional probability independence. Naïve Bayes algorithms differ in assumptions pertaining to probability distribution [4].

Some commonly employed NB algorithms available in PySparkML include Bernoulli, Multinomial, and Gaussian. Also available is an implementation of Complement NB, which is an adapted Multinomial algorithm [6].

Since the NB models cater somewhat to different applications, and are fast to train, our team chose to evaluate three (3) on the project (the intent was primarily to provide a good baseline/competition to the Logistic Regression model, and secondarily to find a better overall model):

- Multinomial NB (MNB) – typical for document classification
- Gaussian NB (GNB) – supports continuous data, i.e., IDF values
- Complement NB (CNB) – can provide more stable results than Multinomial

While NB classifier performance is considered “relatively robust” with respect to parameter selection [4], we chose the “smoothing” parameter (α) for grid search cross validation evaluation. Similar to the Logistic Regression regParam (λ) for regularization, the NB smoothing parameter can reduce overfitting (i.e.,

increasing smoothing (α) reduces model complexity). Finding an optimal amount of smoothing will improve model performance and lends to comparison with the work done with the Logistic Regression model.

Considering the NB model is being generated primarily as a comparison/baseline, the following sections will be somewhat less on detail – e.g., as can be seen from our notebook, we started directly with grid search and forewent the somewhat trivial pre-cross-validation training data fitting and predicting.

NB Model Cross Validation via Grid Search

Table 10: NB Grid Search Parameters

smoothing (α)
0.0
0.2
0.4
0.6
0.8
1.0

Each of the three NB models selected were evaluated with the smoothing parameter set above. Also, as with Logistic Regression grid search, the “numFolds” parameter was set to five (5). From the table and heatmap figure below, we can see the best parameter pairing.

Table 11: NB Grid Search Results (average f1-Scores)

modelType	complement	gaussian	multinomial
smoothing			
0.0	0.390487	0.448804	0.398867
0.2	0.504964	0.448804	0.479376
0.4	0.496831	0.448804	0.481538
0.6	0.492777	0.448804	0.479711
0.8	0.490420	0.448804	0.481114
1.0	0.484371	0.448804	0.478268

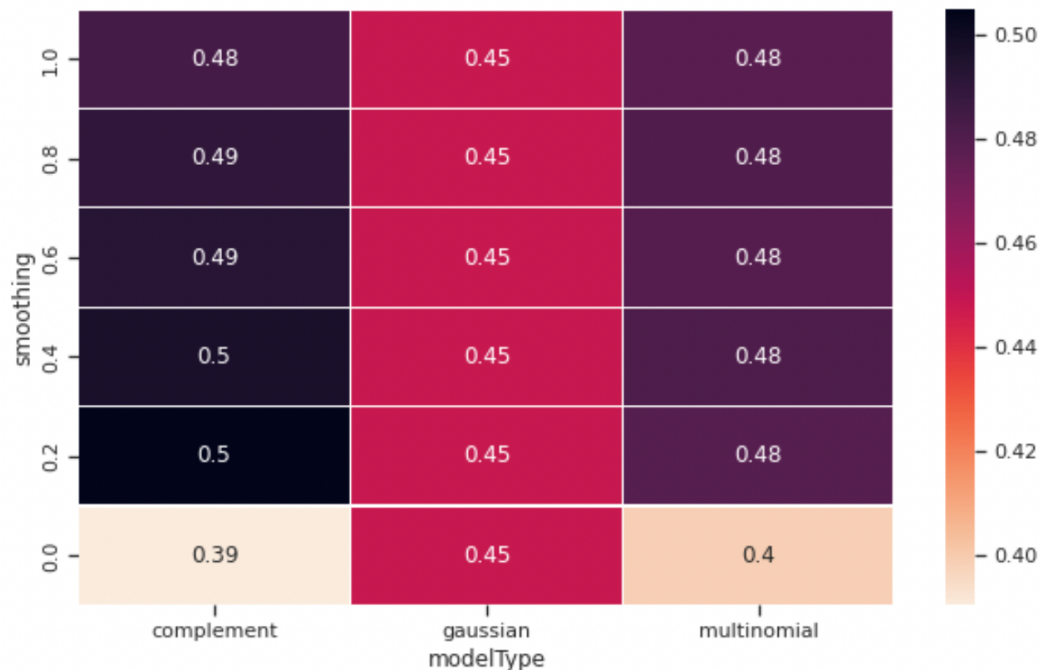


Figure 17: Heatmap – Naïve Bayes Grid Search Results (average f1 scores)

From the tabular data and heatmap above we observe that the best NB classifier algorithm and smoothing parameter combination is: CNB ($\alpha=0.2$). (Also of note are the equal GNB classifier scores – this anomaly may be related to the GNB normal/Gaussian distribution model basis and requires further investigation in a future project phase.)

e) 2) Model Performance

i. / ii. Approach & Results -- Naïve Bayes classifiers

Model Fitting and Prediction: Test Data – best parameters (Case NB-1)

After finding the best hyperparameters of interest, the best model (which is also found via the grid search cross validation) was employed to make predictions on the test set. Refer to confusion matrix figure and case table below.



Figure 18: – Confusion Matrix – Best Model Predictions w/ Test Data (**Case NB-1**)

Based on the results from the confusion matrix above, the False Negative (FN), False Positive (FP), True Negative (TN), and True Positive (TP) have been shown for each class in the table below. This considered the perspective from each class individually, taking each class as their own positive class, and all other classes as the negative class.

Table 12: Confusion Table – Best Model Predictions w/ Test Data (**Case NB-1**)

	Other	Support	Bug	Feature
FN	51	16	25	7
FP	18	72	6	3
TN	100	46	168	190
TP	32	67	2	1

Table 13: Best Model Predictions w/ Test Data (**Case NB-1**)

	Variable	Value
Dataset	Test	-
Model	ComplementNB()	-
Params	smoothing	0.2
Metrics	Accuracy	0.507
	F1 Score	0.470

ComplementNB Classifier Metrics (best parameters - Case NB-1)

Performance metrics for the best parameter Naïve Bayes model are summarized in this section. Individual metric-based conditional formatting has been used in the summary table below to highlight the performing vs. non-performing areas.

Overall Accuracy (total correct label predictions / total target labels) = **0.507**

Table 14: Best Model w/ Test Data: Performance Metrics (Case NB-1)

	Other	Support	Bug	Feature
Recall	0.386	0.807	0.074	0.125
Precision	0.640	0.482	0.250	0.250
f1 Score	0.481	0.604	0.114	0.167

As can be observed, the model is in general providing reasonable performance for “Other” and “Support” labels, while poorly performing with respect to “Bug” and “Feature” labels.

f) MIS-CLASSIFIED ISSUES LABELING

i. Approach

Using the prediction results from the best model, Logistic Regression ($\lambda=0.1, \alpha=0.0$), we have extracted the misclassified labels from the test data set – there are 95 misclassified labels in total.

The team reviewed the provided reference paper [7], which served as a useful reference in our misclassification reason-labeling effort. Following review and a brainstorming session, the team settled on 4 custom reason-labels suited to our project and data:

Insufficient Sample Data: This reason label will be applied to misclassified issues where there was a particular dearth of data.

- Example: All “Feature” issue misclassifications. “Feature” data issues comprised approximately 4% of our full dataset – and after the 80/20 data split, the model only had about 30 “Feature” issues for training.

Imbalanced Data: This reason will be used to tag issues in classes that are under represented in the data set.

- Example: All “Feature” and “Bug” issue misclassifications. “Bug” features only comprise approximately 12% of our full dataset, versus ca 40% for both “Support” and “Other”.

Issue Uncorrelated: This label will be attached to issues that have a title and/or body that does not correlate well with the outcome-based label.

- Example: If an issue is written incoherently or in a foreign language, or about an entirely unrelated subject (e.g. different 3rd-party library).

Input Insufficient:

- Example: Issue title and/or body is written with very few words; or, too few important words such that post the pre-processing pipeline, there will be insufficient features with material weighting (idf) to make a good prediction.

ii. Results

We have assigned misclassification reason-labels on a best assessment basis – i.e., our assignments are based on what we estimate to be the primary / most likely reason. However, most label misclassifications are not obvious and perhaps only via deep inspection of the feature matrix with misclassified issues mappings would allow us to understand why certain issues were misclassified. Further, in many cases, we have included likely secondary reason labels since there may not just be a sole reason (e.g., all misclassifications could be partially attributed to a lack of sample data).

The table below summarizes only the expected primary reasons for misclassification. Please refer to Appendix-A for the detailed issue misclassification labeling spreadsheet (note: some issues were too big to display in the log table cells).

Table 15: Misclassified Issues – Primary Reason Labels Summary

	Bug	Feature	Other	Support	Sub-Totals	
Insufficient Sample Data	1	6	14	15	36	38%
Input Insufficient	15	1	12	22	50	53%
Imbalanced Data	8	0	0	0	8	8%
Issue Uncorrelated	0	0	1	0	1	1%
Sub-Totals	24	7	27	37	95	100%
	25%	7%	28%	39%		

Notable observations from the summary table above include:

- With respect to reason-labels, misclassifications appear concentrated in the “Insufficient Sample Data” and “Input Insufficient” categories.
- In terms of target labels, the misclassifications were mainly spread between Bug, Other, and Support, with the latter being most erroneous.
- Feature misclassifications appear low because they are under-represented in the data.

Insufficiency of data appears to be main problem driving label misclassification, which is something the project team had postulated. In a future project phase, further data collection would be necessary. Additional samples may somewhat address the imbalance issue as well. Poor/insufficient user issue inputs may always be a problem and will likely not be solved by a larger data set.

4) DISCUSSION

a) CLASSIFIER MODEL COMPARISON

Overall, both the Logistic Regression and ComplementNB models performed relatively poorly, with the former slightly outperforming.

The above statement is accurate if looking from the point of view of having an implementable/commercial-ready model; however, from the standpoint of this project, which might constitute a preliminary project phase in research or industry, the models could be considered as performing reasonably – especially accounting for the small and imbalanced dataset utilized, lack of domain knowledge, etc.

Table 16: Best Models Comparison – Overall Performance Comparison

		LR ($\lambda=0.1, \alpha=0.0$)	CNB ($\alpha=0.2$)
	Variable	Value	Value
Metrics	Accuracy	0.527	0.507
	F1 Score	0.500	0.470

Table 17: Best Models Comparison – Label-based Performance Metrics (Differences)

	Other	Support	Bug	Feature
Recall	0.288	-0.253	0.037	0.000
Precision	-0.126	0.053	0.500	0.250
f1 Score	0.102	-0.060	0.080	0.033

It can be observed from the label-based metric differences that, except for “Support” label precision, the Logistic Regression model generally matches or exceeds the performance of the ComplementNB (CNB) model. Light green and yellow in the above table show areas where the CNB model is competing, or outperforming (i.e., red text), the Logistic Regression model.

b) REAL WORLD SCENARIOS

Graeme Folk:

Automating GitHub issue classification can be useful in many scenarios, such as when a project receives many issues, and it is not feasible for a team of developers to manually review and classify each issue. By automating this process, developers can save time and effort, and can focus on addressing the issues that require their attention.

Stephen Thistle:

In a future project phase (post extensive testing and validation), should our GitHub issue classification algorithm reach high levels of performance, it may be able to be extended to wider application (i.e. screening and prioritizing different issues, messages, or communications). Retraining and re-validation

would be needed, of course, but perhaps some transfer learning could be realized. Examples of potential non-safety non-critical applications that may be appropriate, include other github-like issue repositories, customer feedback email categorization, or chat forum screening.

David Cheng:

Additionally, automating GitHub issue classification can also be helpful in ensuring that issues are assigned to the appropriate developers. By automatically assigning issues to the correct developers based on the issue type, developers can focus on the issues that they are best equipped to handle and can avoid wasting time on issues that are outside of their expertise.

Overall, automating GitHub issue classification can help improve the efficiency and effectiveness of a development team, and can help ensure that issues are properly addressed and resolved in a timely manner.

5) CONCLUSION

In this paper, we have proposed a method for automating the classification of GitHub issues using machine learning. Our approach involves training a logistic regression model and Naïve Bayes model on a dataset of labeled issues taking in the “title”, “body”, and “author_association” metadata for preprocessing. The trained models were used to automatically classify novel issues as they are reported on a project, helping project maintainers to prioritize and address them more effectively. This method showed promising results with comparable results in both Naïve Bayes and Logistic Regression accuracies when classifying over the validation dataset.

Further experimentation with different model types and parameters may be necessary to determine the optimal approach. There are a few different options that can be considered to improve the performance of the proposed method for automating the classification of GitHub issues. For example, the model could be trained on a larger dataset of labeled issues to improve its accuracy and capture a wider range of characteristics and variations. Additional types of machine learning models could be tested and compared to determine the best approach for this task. It may also be useful to incorporate additional features or information into the model to provide additional context and improve the model's ability to classify the issue accurately. Overall, automating the classification of GitHub issues can help project maintainers manage their workload more efficiently and improve the overall quality of the project

6) REFERENCES

- [1] Fragkiskos Chatziasimidis, Ioannis Stamelos. Data collection and analysis of GitHub repositories and users. In 6th International Conference on Information, Intelligence, Systems and Applications (IISA), page 1, 2015
- [2] Muller, A. C., Guido, S. (2017). "Introduction to Machine Learning with Python – A Guide for Data Scientists". O'Reilly Media, Inc. 978-1-449-36941-5.
- [3] H. Alhammady, "Weighted Naive Bayesian Classifier," 2007 IEEE/ACS International Conference on Computer Systems and Applications, 2007, pp. 437-441, doi: 10.1109/AICCSA.2007.370918.
- [4] "1.9. Naive Bayes", scikit learn, 2022. https://scikitlearn.org/stable/modules/naive_bayes.html
- [5] "10. Regularization", Learning Apache Spark with Python, 2022. <https://runawayhorse001.github.io/LearningApacheSpark/reg.html>
- [6] "Naive Bayes", Apache Spark, 2022. <https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.classification.NaiveBayes.html>
- [7] N. Novielli, D. Girardi and F. Lanubile 2018. "A Benchmark Study on Sentiment Analysis for Software Engineering Research". In Proceedings of 15th International Conference on Mining Software Repositories (MSR 2018), May 28 - 29, 2018 Gothenburg, Sweden, 12 pages. DOI: 10.1145/3196398.3196403

APPENDIX A – MISCLASSIFIED ISSUES LIST & REASON LABELS

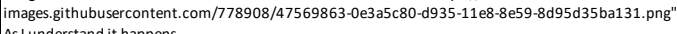
number	prediction	PredictionLabel	number	title	author_association	body	Target	Likely Primary Misclassification Reason	Possible Secondary Reason(s)
12012	0	Other	12012	React.Children.toArray and React.cloneElement do not work with portal elements	NONE	<!-- Note: if the issue is about documentation or the website, please file it at: https://github.com/reactjs/reactjs.org/issues/new --> **Do you want to request a *feature* or report a *bug*?** BUG or undefined behaviour **What is the current behavior?** Doing ... React.Children.toArray(ReactDOM.createPortal(...)) ... fails with: ... Objects are not valid as a React child (found: object with keys {\$\$typeof, key, children, containerInfo, implementation}). If you meant to render a collection of children, use an array instead. ... Namely, the following complete snippet fails: ```jsx	Feature	Insufficient Sample Data	Imbalanced Data
12197	1	Support	12197	Warn on ComponentName.PropTypes	CONTRIBUTOR	<!-- Note: if the issue is about documentation or the website, please file it at: https://github.com/reactjs/reactjs.org/issues/new --> **Do you want to request a *feature* or report a *bug*?** Feature **What is the current behavior?** Sometimes, when my boss is stressing me out, I add prop types this way: ```js const MyButton = props => <button>/* some fancy implementation */</button>; MyButton.propTypes = { children: node.isRequired }; ... Notice the upper-case P in PropTypes. React does not warn me about this typo. **What is the expected behavior?** I'd love to get a warning about this (and 'DefaultProps' for that sake) whenever I mis-type them. I would actually love to implement a pull request for this, but I couldn't find anywhere else where you're warning about static properties like these. Where would you like me to put this code?	Other	Issue Uncorrelated	
12238	0	Other	12238	New React component definition with autobind	NONE	I don't know what exactly is planned, but auto bind for methods would be nice. Yeah, I know property initializers, still, a user should not have to think about that. I suppose a state should be stored in an object, not in a closure because of logs, and component should be an object ideally. Nevermind, feel free to close this "issue" without any explanation, I am just curious the reasoning behind a new design.	Support	Input Insufficient	Insufficient Sample Data

12301	0	Other	12301	state change in td element	NONE	<p>Displaying tabular rows, when setting data in td element, on state change it displays only the currently updated td element data..all the other ones go missing from screen. When the same thing is placed within text box in each td element things work fine.</p> <p>Following is sample the code :</p> <p>Working code : `<tr key={id}>` `<td> <input value={this.state.price[id]}> </td>`</p> <p>Failing code : `<tr key={id}>` `<td> {this.state.price[id]} </td>`</p>	Support	Input Insufficient	Insufficient Sample Data
12340	0	Other	12340	jsx's close tag name may optional?	NONE	<p>**Do you want to request a *feature* or report a *bug*?*</p> <p>*feature*</p> <p>**What is the current behavior?** <pre> `html <div> <p>some text</p> <hr/> </div> ` </pre></p> <p>**What is the expected behavior?** because the html tag always pairs, so if simple write better? <pre> `html <div> <p>some text</> <hr/> </> ` </pre></p>	Feature	Input Insufficient	Imbalanced Data, Insufficient Sample Data
12372	1	Support	12372	react 16.3 new context API basic intent	NONE	<pre> <!-- Note: if the issue is about documentation or the website, please file it at: https://github.com/reactjs/reactjs.org/issues/new --> **Do you want to request a *feature* or report a *bug*?* misunderstanding **What is the current behavior?** Impossible to render a component which is a provider furnished by createContext function trying to follow those articles : [whats new in react](https://medium.com/@baphemot/whats-new-in-react-16-3-d2c9b7b6193b) [react new context API](https://medium.com/dailys/reacts-%EF%B8%8F-new-context-api-70c9fe01596b) **If the current behavior is a bug, please provide the steps to reproduce and if possible a minimal demo of the problem. Your bug will get fixed much faster if we can run your code and it doesn't have dependencies other than React. Paste the link to your JSFiddle (https://jsfiddle.net/Luktwrnm/) or CodeSandbox (https://codesandbox.io/s/new) example below:** [code sandBox example](https://codesandbox.io/embed/k591mqw98o) sourceCode of codeSandBox : ` import React, { Component, createContext } from "react"; import { render } from "react-dom"; const CartState = createContext({ open: false </pre>	Other	Input Insufficient	

12485	1	Support	12485	Error still logged to console when caught by error boundary in testing	NONE	<p>**Do you want to request a *feature* or report a *bug*?**</p> <p>I'm not quite sure if feature/bug... I have an error boundary, and I want to test that it works correctly, but it produces a big 'console.error' and there doesn't seem a great way to get around it.</p> <p>**What is the current behavior?**</p> <p>When an error is captured by the error boundary, it still logs to console during tests, making it look like something failed but it is expected (because I am testing error boundary) in this case.</p> <p>...</p> <pre>console.error node_modules/react-dom/cjs/react-dom.development.js:9747 The above error occurred in the <Child> component: in Child (at ErrorBoundary.test.jsx:20) in ErrorBoundary (created by WrapperComponent) in WrapperComponent</pre> <p>React will try to recreate this component tree from scratch using the error boundary you provided, ErrorBoundary.</p> <p>...</p> <p>**If the current behavior is a bug, please provide the steps to reproduce and if possible a minimal demo of the problem. Your bug will get fixed much faster if we can run your code and it doesn't have dependencies other than React. Paste the link to your JSFiddle (https://jsfiddle.net/Luktwrmd/) or CodeSandbox (https://codesandbox.io/s/new) example below:**</p> <p>Error Boundary:</p>	Other	Input Insufficient	
12551	1	Support	12551	New Context Provider may block Old context propagation if children are constant	CONTRIBUTOR	<p><!-- Note: if the issue is about documentation or the website, please file it at: https://github.com/reactjs/reactjs.org/issues/new --></p> <p>**Do you want to request a *feature* or report a *bug*?**</p> <p>Bug</p> <p>**What is the current behavior?**</p> <p>It seems that, if the children of a new-style 'React.createContext()' context 'Provider' are constant, the 'Provider' can block updates from old-style 'this.context' context providers from propagating to 'this.context' consumers.</p> <p>This sandbox demonstrates the issue. Clicking the button with a number will correctly increment the 'Root's 'state' and 'context', but the update is only propagated to the 'Child3's 'context' (and its button) when the "Colors!" button is clicked, as it causes an update to the 'value' of the new-style 'Provider':</p> <p>https://codesandbox.io/s/ol4lpokpjy</p> <p><details> <summary>Copy of the source code in the sandbox</summary></p> <pre>```jsx import PropTypes from "prop-types"; import React from "react"; import ReactDOM from "react-dom";</pre>	Bug	Insufficient Sample Data	Imbalanced Data, Input Insufficient

12693	1	Support	12693	react-test-renderer.create does not work properly with forwardRef components	NONE	<p>**Do you want to request a *feature* or report a *bug*?**</p> <p>Bug</p> <p>**What is the current behavior?*</p> <p>If you try to access the <code>.root`</code> of a component tree of a component created with <code>forwardRef</code>, you will get an error</p> <pre>> `Unexpected object passed to ReactTestInstance constructor`</pre> <p>**If the current behavior is a bug, please provide the steps to reproduce and if possible a minimal demo of the problem. Your bug will get fixed much faster if we can run your code and it doesn't have dependencies other than React. Paste the link to your JSFiddle (https://jsfiddle.net/Luktwrmd/) or CodeSandbox (https://codesandbox.io/s/new) example below:**</p> <p>A minimal example is simply a component created with <code>forwardRef</code>:</p> <pre>'''js import React from "react"; export default React.forwardRef(() => <div>hello</div>); '''</pre> <p>Now if you try to use <code>create`</code> from <code>react-test-renderer`</code> and access <code>.root`</code>, you will get</p> <pre>> Unexpected object passed to ReactTestInstance constructor (tag: 14). This is probably a bug in React.</pre> <pre>'''js import React from "react";</pre>	Bug	Imbalanced Data	Insufficient Data
12723	0	Other	12723	The findByType method from react-test-renderer doesn't find ref-forwarding components	CONTRIBUTOR	<p>**Do you want to request a *feature* or report a *bug*?**</p> <p>Bug</p> <p>**What is the current behavior?*</p> <p>In tests, if the component passed to <code>root.findByType`</code> was created using <code>forwardRef`</code> it can't be found.</p> <p>If I use <code>root.find`</code> to loop over all the nodes and log <code>.type`</code> I see that the ref-forwarding-component is skipped entirely.</p> <p>**If the current behavior is a bug, please provide the steps to reproduce and if possible a minimal demo of the problem. Your bug will get fixed much faster if we can run your code and it doesn't have dependencies other than React. Paste the link to your JSFiddle (https://jsfiddle.net/Luktwrmd/) or CodeSandbox (https://codesandbox.io/s/new) example below:**</p> <pre>https://codesandbox.io/s/k0o70vjv07?module=%2Fsrc%2Findex.test.js</pre> <p>I expect the second test to work like the first, however the second test is failing.</p> <p>**What is the expected behavior?*</p> <p>This may just be the way that <code>forwardRef`</code> works, but the reason I expected it to work is I'm using it to make wrapped controls, like the <code><FancyButton>`</code> example from the documentation. When writing tests I want to be able to find a <code>FancyButton`</code> and test it's being passed the correct props, and not have to check that the DOM rendered by <code>FancyButton`</code> is correct; that's already being tested in <code>fancy-button.test.js`</code></p> <p>**Which versions of React, and which browser / OS are affected by this issue? Did this work in previous versions of React?*</p> <p>Using React 16.3.2, failing in Chrome and Node 8.11.0</p>	Bug	Imbalanced Data	Insufficient Data
12774	0	Other	12774	this.state doesn't get the current state properly	NONE	<p>Related to this issue</p> <pre>https://github.com/dmtrKovalenko/material-ui-pickers/issues/396#issuecomment-387972773</pre> <p>I found a work around by checking the state out of the <code>onChange`</code>. But i can't figure out why it behaves like that in the first place.</p> <p>Example:</p> <pre>https://codesandbox.io/s/2zrx49zwjj</pre>	Support	Input Insufficient	Insufficient Sample Data

13410	1	Support	13410	Investigate why select._wrapperState.initialVa lue is necessary	CONTRIBUTOR	Uncovered during @raunofreiberg's select work (#13389). 'select._wrapperState.initialValue' is only ever assigned, never referenced. Can it be removed? Example: https://github.com/facebook/react/blob/master/packages/react-dom/src/client/ReactDOMFiberSelect.js#L191 It would be great if someone could investigate this, and figure out if it can be removed.	Bug	Imbalanced Data	Insufficient Sample Data, Input Insufficient
13715	0	Other	13715	Improved stack trace	NONE	**Do you want to request a *feature* or report a *bug*?** *feature* **What is the current behavior?** *The usability of the current stack trace for components with native dom elements can be improved. ![image](https://user-images.githubusercontent.com/43165983/45943746-33217400-c005-11e8-953b-2e016bd10f9d.png) *In this example it tells me that there are 3 divs, followed by a select tag (Not sure why select's display name is showing as the components' name itself) and then option tag. But this same structure exists in lot of places in the component. **What is the expected behavior?** *Stack trace with line number/some other way to pinpoint it's location (like id/class if provided??) is desired **Which versions of React, and which browser / OS are affected by this issue? Did this work in previous versions of React?* OS: Ubuntu 18.04 React: 16.2.0	Support	Input Insufficient	Insufficient Sample Data
13819	1	Support	13819	How to send a notification to specific user and handle it	NONE	I'm an android and iOS developer. on android and ios programming I can open a ***specific page*** when user click on the notification. for example, this is a sample JSON string from the server side to clients: "data":{ "pages":"home" } when I receive this notification I can open the 'home' page. I want to handle my react pages when a notification will receive. Is it possible to handle this action? or on the web application, user just can see the message from notification?	Other	Input Insufficient	
13845	0	Other	13845	[npm:create-react-class] Forwarding refs	NONE	https://reactjs.org/docs/react-without-es6.html doesn't seem to explain how to use forwarding refs: https://reactjs.org/docs/forwarding-refs.html Does the render function in the class have to have the new props argument?	Support	Input Insufficient	Insufficient Sample Data

13944	0	Other	13944	Cannot be used new contextType API in constructor?	NONE	<p><!-- Note: if the issue is about documentation or the website, please file it at: https://github.com/reactjs/reactjs.org/issues/new --></p> <p>**Do you want to request a *feature* or report a *bug*? ** *feature*</p> <p>**What is the current behavior? ** Using `Component.contextType`, `this.context` keeps `null` in class component's constructor.</p> <p>**What is the expected behavior? **</p> <p>- `this.context` can be obtained in constructor - if implementing of the feature is inappropriate, write in documentations that `this.context` cannot be used in constructor.</p> <p>**Which versions of React, and which browser / OS are affected by this issue? Did this work in previous versions of React? ** React 16.6 (older React doesn't have `Component.contextType`) in Chrome 70 (Windows 7)</p>	Bug	Input Insufficient	
13985	1	Support	13985	Warn if ReactDOM.createPortal is inside a noscript or other text content tag	CONTRIBUTOR	<p>**Do you want to request a *feature* or report a *bug*? ** bug</p> <p>**What is the current behavior? **</p> <p>When calling `ReactDOM.createPortal` from within a `<noscript >`,="" error="" happens,="" is="" logged.<="" no="" nothing="" p=""> <pre> '''javascript ReactDOM.render(<noscript> {ReactDOM.createPortal("yo", modalNode)} </noscript>, appNode); ''' </pre> <p>**What is the expected behavior? **</p> <p>Whether:</p> <p>- warn that it doesn't work (it worked in the previous version) - render the portal</p> <p>**Which versions of React, and which browser / OS are affected by this issue? Did this work in previous versions of React? **</p> <p>- React 16.5.0 & more recent - Reproducible in every browser</p> </noscript></p>	Feature	Insufficient Sample Data	Imbalanced Data
14002	0	Other	14002	Safari Devtools flooded with security errors on react-dom selection work with iframes with diff origins	NONE	<p>hey folks, looks like [this code](https://github.com/facebook/react/commit/b565f495319750d98628425d120312997bee410b) added small issue with safari and it could flood devtools console output with messages like this:</p> <p></p> <p>As I understand it happens [here](https://github.com/facebook/react/commit/b565f495319750d98628425d120312997bee410b#diff-a654f37b01573fc8006b426d56ad53ceR50) and I see you catch the error, but safari still adds the error message if you have iframes with different origin</p>	Bug	Input Insufficient	Insufficient Sample Data, Imbalanced Data

14009	0	Other	14009	React + GSAP -> Animation doesn't work with Build production	NONE	<p>#### Is this a bug report? - Maybe Yes? A doubt :)</p> <p>#### Did you try recovering your dependencies? - Yes</p> <p>#### Which terms did you search for in User Guide? - Yarn Build: Animation Failing</p> <p>#### Environment</p> <p>`npx create-react-app gsapApp`</p> <p>**package.json**</p> <pre> { "name": "gsapApp", "version": "0.1.0", "private": true, "scripts": { "start": "react-scripts start", "build": "react-scripts build", "test": "react-scripts test", "eject": "react-scripts eject" }, "dependencies": { "gsap": "^2.0.2", "react": "^16.6.0", "react-dom": "^16.6.0", "react-scripts": "^2.0.5" }, "devDependencies": {} } </pre>	Support	Insufficient Sample Data	Input Insufficient
14160	0	Other	14160	XSS Protection: href Object not being html escaped as a props	NONE	<p><!-- Note: if the issue is about documentation or the website, please file it at: https://github.com/reactjs/reactjs.org/issues/new --></p> <p>**Do you want to request a *feature* or report a *bug*?** I would like to request a bug.</p> <p>**What is the current behavior?** From what I know, it is possible to inject object in props. However, this object seems to be html escaped when inserted into the DOM from my observation.</p> <p>Thus, if I try to add an <code>onerror=alert('XSS')</code> in a <code></code> tag through a props, this is gonna be escaped when rendered. Then, I realized that inserting an <code>id='test'</code> is totally possible with a props. So I thought only dangerous javascript injectable attributes are escaped such as <code>onerror</code>, <code>onload</code>, ...</p> <p>However, I realized that the <code>href=javascript:alert('1')</code> is not escaped when inserted through a props. The javascript gets executed. Here, I thought it might a bug.</p> <p>**If the current behavior is a bug, please provide the steps to reproduce and if possible a minimal demo of the problem. Your bug will get fixed much faster if we can run your code and it doesn't have dependencies other than React. Paste the link to your JSFiddle (https://jsfiddle.net/Luktwrmd/) or CodeSandbox (https://codesandbox.io/s/new) example below:**</p> <pre> class App extends Component { render() { </pre>	Support	Insufficient Sample Data	Input Insufficient

14266	0	Other	14266	props with same key and value	NONE	<p><!-- Note: if the issue is about documentation or the website, please file it at: https://github.com/reactjs/reactjs.org/issues/new --></p> <p>**Short form for props key and value** for example: *Instead of this*: '''Javascript const { name, age, photoURI } = this.state return (<Person name={name} age={age} photoURI={photoURI} />) ''' this: '''Javascript const { name, age, photoURI } = this.state return (<Person *name *age *photoURI />) '''</p> <p>or even other way so that we don't have to write same key and value.</p>	Support	Insufficient Sample Data	Input Insufficient
14355	0	Other	14355	[Mouse Events] onMouseDownCapture is not stopping capture phase propagation	NONE	<p>**Bug report**</p> <p>**The current documentation has and I quote https://reactjs.org/docs/events.html ** **To register an event handler for the capture phase, append Capture to the event name; for example, instead of using onClick, you would use onClickCapture.**</p> <p>**I tried with this https://jsbin.com/hilome/edit?js,output and**</p> <pre>`<div id="grandparent" onClickCapture={this._handleClickCapture}> <div id="parent" onClick={this._handleClickBubble}> <button id="elem" onClick={this._handleClick}>Click me!</button> </div> </div>`</pre> <p>**if I change onClickCapture with onMouseDownCapture the stopPropagation() method does not prevent capture phase any more**</p> <p>**What is the expected behavior?**</p> <p>**I would expect that stopPropagation() stops the propagation to its children as it works with onClickCapture event**</p>	Support	Input Insufficient	Insufficient Sample Data
14496	0	Other	14496	Local environment is not a problem after the line Error: Minified React error #31; visit	NONE	<p>react-dom.production.min.js:179 Error: Minified React error #31; visit https://reactjs.org/docs/error-decoder.html?invariant=31&args[]=object%20with%20keys%20%7Bmodule%2C%20planList%2C%20taskList%7D&args[]=for the full message or use the non-minified dev environment for full errors and additional helpful warnings.</p>	Support	Input Insufficient	Insufficient Sample Data
14537	1	Support	14537	Suggestion: make version of react only with hooks to reduce bundle size	NONE	<p>I think that should be a version of react without component stuff to reduce bundle size and another version with component and hooks Is that would make a big effect of the bundle size or they are just some KBs ?</p>	Other	Input Insufficient	Insufficient Sample Data

14544	1	Support	14544	'yarn flow' stuck at merging on Window10	CONTRIBUTOR	<p>**Do you want to request a *feature* or report a *bug*?* a bug</p> <p>**What is the current behavior?* See #14519 first.</p> <p>And i tested on a Linux machine,same node version,same yarn version,same npm version,and same operation. On Linux,'yarn flow' works well. I tested on another Window10 machine,also fail.</p> <p>So i think this is a **bug** related to flow,win10 and react source code,not my own problem.</p>	Other	Input Insufficient	Insufficient Sample Data
14731	1	Support	14731	useState inside a context provider not properly read when called from timeout	NONE	<p><!-- Note: if the issue is about documentation or the website, please file it at: https://github.com/reactjs/reactjs.org/issues/new --></p> <p>**Do you want to request a *feature* or report a *bug*?*</p> <p>Looks like a **bug**.</p> <p>**What is the current behavior?*</p> <p>I have a simple context set up to manage a global store. When implementing this context as a functional component with the useState hook, calls to my setStore function from inside a timeout are seeing old versions of the store and updating it incorrectly.</p> <p>Possibly a duplicate of #14010 but I don't see why the value of my store should be getting saved by the closure. The closure created by the setTimeout can't see the value of store, so it shouldn't be captured.</p> <p>**If the current behavior is a bug, please provide the steps to reproduce and if possible a minimal demo of the problem. Your bug will get fixed much faster if we can run your code and it doesn't have dependencies other than React. Paste the link to your JSFiddle (https://jsfiddle.net/Luktwrmd/) or CodeSandbox (https://codesandbox.io/s/new) example below:*</p> <p>https://codesandbox.io/s/mnxr754oy</p> <p>Refresh the page in the sandbox and click the "Increment otherVal" button a few times. After 3 seconds, a timeout fires in ChildThree that sets myVal to 42 but doesn't touch otherVal; however, the changes made by incrementing</p>	Other	Insufficient Sample Data	Input Insufficient
14856	0	Other	14856	Chrome 73 breaks wheel events	NONE	<p>Similar to #8968, but for the 'wheel' and 'mousewheel' events. They are now passive by default for root elements in Chrome 73 (currently beta) which means React apps that have custom scrolling/zooming behaviors will run into issues.</p> <p>The quick fix may be to manually add event listeners with `{passive: false}` but has the React team considered if this should be configurable for the React event handler?</p> <p>Blog post from the Chrome team here: https://developers.google.com/web/updates/2019/02/scrolling-intervention</p>	Bug	Input Insufficient	Insufficient Sample Data, imbalanced Data

14904	1	Support	14904	controlled input cursor jumps to end (again)	NONE	<p><!-- Note: if the issue is about documentation or the website, please file it at: https://github.com/reactjs/reactjs.org/issues/new --></p> <p>**Do you want to request a *feature* or report a *bug*?**</p> <p>Bug</p> <p>**What is the current behavior?**</p> <p>when typing in a controlled input, the cursor always jumps to the end. This was an old issue that seems to have resurfaced.</p> <p>[this code pen](https://codepen.io/gaearon/pen/VmmPgp?editors=0010) used in the docs [here](https://reactjs.org/docs/forms.html#controlled-components) has the problem in all browsers as far as I have been able to test.</p> <p>**What is the expected behavior?**</p> <p>because we are using the state to update the component as soon as it's changed, the input element should be able to keep the cursor in the same place.</p> <p>**Which versions of React, and which browser / OS are affected by this issue? Did this work in previous versions of React?*</p> <p>I'm at latest (16.8.2) and I tested on Chrome, FireFox, and Edge on Windows</p>	Bug	Input Insufficient	Insufficient Sample Data, imbalanced Data
14946	0	Other	14946	Feature request: 'useShouldUpdate' hook	NONE	<p>**Do you want to request a *feature* or report a *bug*?**</p> <p>Request a feature</p> <p>It is currently impossible to let React know that the output of a **functional component** is not affected by the current change. I think that it would be great to have a hook for enabling this capability. In other words: having the SCU functionality in functional components through a hook.</p> <p>I'm aware of the existence of 'React.memo' and I know that it is possible to accomplish the same thing by splitting the logic in 2 different functional components... as long as we enhance the "base" component with 'React.memo'. However, I still think that the hook that I'm suggesting would be a pretty nice addition.</p> <p>I guess that it's a good idea to show an example of a real case where this hook would be helpful. So, here we go:</p> <p>I don't normally use the official react-redux bindings. Instead, I have my own version of the 'connect' HOC which has a more limited (and slightly different) API, that better suits my needs. This makes my version a bit more performant and a lot lighter. Regardless of whether it is a good idea not to use the official react-redux bindings, the following example illustrates the benefit of having the hook that I'm suggesting.</p> <p>This is the current implementation of my custom "connect" HOC:</p> <pre> '''js const emptyObj = {}; const alwaysEmpty = () => emptyObj; export default (fromStateProps_, fromActionProps = emptyObj, mapper) => { const dependsOnProps = fromStateProps_ && fromStateProps_.length !== 1; const fromStateProps = !fromStateProps_ ? alwaysEmpty : fromStateProps_; </pre>	Support	Insufficient Sample Data	Input Insufficient
14956	1	Support	14956	Element.createShadowRoot is deprecated and will be removed in M73, around March 2019. Please use Element.attachShadow instead	NONE	<p>include.preload.js:1 [Deprecation] Element.createShadowRoot is deprecated and will be removed in M73, around March 2019. Please use Element.attachShadow instead. See https://www.chromestatus.com/features/4507242028072960 for more details.</p> <p>**What is the expected behavior?**</p> <p>**Which versions of React, and which browser / OS are affected by this issue? Did this work in previous versions of React?*</p>	Other	Input Insufficient	Insufficient Sample Data

15054	0	Other	15054	useImperativeHandle callback never called (when rendering w/ enzyme)	NONE	<p>**Do you want to request a *feature* or report a *bug*?** bug</p> <p>**What is the current behavior?**</p> <p>I have the code:</p> <pre> ... function Form(props, ref) { React.useImperativeHandle(ref, () => { debugger; return { setErrors: () => {}, }; }); } export default React.forwardRef(Form); ... </pre> <p>When I use the component, the callback passed to `useImperativeHandle` is never called. (The debugger statement is never hit).</p> <p>The code that I have using the component is:</p> <pre> ... </pre>	Support	Insufficient Sample Data	Input Insufficient
15056	1	Support	15056	Controlled contentEditable element contents not updated when state changes	NONE	<p>**Do you want to request a *feature* or report a *bug*?** Bug</p> <p>**What is the current behavior?** An element with contentEditable does not update the content when state updates.</p> <p>Reproduce: https://codesandbox.io/s/kk421m2jmr 1. Type something in the contentEditable div. 2. Click reset 3. Expected div to contain "TEXT RESET!"</p> <p>**Which versions of React 16.8.4</p>	Other	Input Insufficient	Insufficient Sample Data
15209	0	Other	15209	useState function as initial state gets executed	NONE	<p>**Do you want to request a *feature* or report a *bug*?** bug</p> <p>**What is the current behavior?** When passing a function as the initial value to the useState hook the function gets executed.</p> <p>Please see the following example: https://codesandbox.io/s/vvj88kqn4l</p> <p>**What is the expected behavior?** The function is not executed.</p> <p>Why does the function get executed here?</p> <p>**Which versions of React, and which browser / OS are affected by this issue? Did this work in previous versions of React?* 16.8.3</p>	Support	Input Insufficient	Insufficient Sample Data

15236	0	Other	15236	How test componentDidUpdate lifecycle method with test-renderer?	NONE	<p><!-- Note: if the issue is about documentation or the website, please file it at: https://github.com/reactjs/reactjs.org/issues/new --></p> <p>**Do you want to request a *feature* or report a *bug*?**</p> <p>**What is the current behavior?**</p> <p>**If the current behavior is a bug, please provide the steps to reproduce and if possible a minimal demo of the problem. Your bug will get fixed much faster if we can run your code and it doesn't have dependencies other than React. Paste the link to your JSFiddle (https://jsfiddle.net/Luktwrdm/) or CodeSandbox (https://codesandbox.io/s/new) example below:**</p> <p>**What is the expected behavior?**</p> <p>**Which versions of React, and which browser / OS are affected by this issue? Did this work in previous versions of React?**</p>	Support	Input Insufficient	Insufficient Sample Data
15513	0	Other	15513	Allow ReactNode as a type for the child of <option/>	NONE	<p>**Do you want to request a *feature* or report a *bug*?** Feature</p> <p>**What is the current behavior?** Currently, the options element only allows types number and string.</p> <p>**What is the expected behavior?** An option should allow for a ReactNode as a child in addition to a number + string.</p> <p>**Which versions of React, and which browser / OS are affected by this issue? Did this work in previous versions of React?** All versions. All browser types. To the best of my knowledge, no.</p> <p>p.s. This is my first feature request here, so let me know if I need to adjust the feature request in any way.</p>	Support	Input Insufficient	Insufficient Sample Data
15527	1	Support	15527	Apparent memory leak using hooks	NONE	<p>**Do you want to request a *feature* or report a *bug*?**</p> <p>Maybe a bug. Maybe I'm just running into a weird edge case.</p> <p>**What is the current behavior?**</p> <p>I have a situation where using a combination of `useEffect`, `useCallback`, and having a function in the actual component render is causing a sort of memory leak. The current render of the component seems to hang on to references of past renders of the component. (more details in the reproduction sample repo)</p> <p>**If the current behavior is a bug, please provide the steps to reproduce and if possible a minimal demo of the problem.**</p> <p>I think I have this minimized about as much as I can: https://github.com/rally25rs/react-mem</p> <p>It's a basic `create-react-app` project, so you can just `yarn install` & `yarn start` it. There is some explanation/instructions in the web page that it renders.</p> <p>The `src/VirtualizedTable.js` file is the key file to look at.</p> <p>**What is the expected behavior?**</p> <p>As best as my brain can comprehend, memoized references should be getting cleared as the component props change and data should be garbage collected (but it isn't).</p>	Other	Insufficient Sample Data	Input Insufficient

15723	1	Support	15723	Strange onScroll behaviour	NONE	<p>**Do you want to request a *feature* or report a *bug*?**</p> <p>Possible bug</p> <p>**What is the current behavior?**</p> <p>onScroll callback on parent element fires when children element is scrolled. Native listener working as expected, though.</p> <p>Example with reproduction https://codesandbox.io/s/kk3th</p> <p>Just try to scroll little box with items.</p> <p>**What is the expected behavior?**</p> <p>I am not sure if this behaviour is correct, but it was unexpected for me, so it might be a bug. I was not expecting onScroll to fire at all.</p> <p>**Which versions of React, and which browser / OS are affected by this issue? Did this work in previous versions of React?**</p> <p>"react": "16.8.6" macOs Mojave 10.14.5</p> <p>Did not tried any other versions</p>	Bug	Input Insufficient	Insufficient Sample Data, imbalanced Data
15856	1	Support	15856	No warning or Error on component mounting itself	NONE	<p><!-- Note: if the issue is about documentation or the website, please file it at: https://github.com/reactjs/reactjs.org/issues/new --></p> <p>**Do you want to request a *feature* or report a *bug*?**</p> <p>bug</p> <p>**What is the current behavior?**</p> <p>When mounting a component within itself, there is no warning or error, and leads to an infinite loop until the browser tab crashes.</p> <p>**If the current behavior is a bug, please provide the steps to reproduce and if possible a minimal demo of the problem. Your bug will get fixed much faster if we can run your code and it doesn't have dependencies other than React. Paste the link to your JSFiddle (https://jsfiddle.net/Luktwrmd/) or CodeSandbox (https://codesandbox.io/s/new) example below:**</p> <p>...</p> <pre>const Card = (<Card>test</Card>);</pre> <p><Card /></p> <p>...</p> <p>**What is the expected behavior?**</p> <p>warning in dev tools or throws an error</p>	Other	Insufficient Sample Data	Input Insufficient

16093	1	Support	16093	onClick not firing	NONE	<p>I can't seem to get the onClick event to fire at all. I've reduced my app to the following file:</p> <pre> index.js: ... import React from 'react' import { render } from 'react-dom' const Index = () => { console.log('rendered') return <button onClick={() => console.log('hello')}>test</button> } render(<Index />, document.getElementById('app')) ... I get 'rendered' output to the console, but nothing when I click on the button. package.json: ... { ... devDependencies { ... "react": "^16.8.4", "react-dom": "^16.8.4", </pre>	Other	Insufficient Sample Data	Input Insufficient
16305	1	Support	16305	Different Suspense Behavior in ReactDOM.render vs React.createRoot().render	NONE	<pre> <!-- Note: if the issue is about documentation or the website, please file it at: https://github.com/reactjs/reactjs.org/issues/new --> **Do you want to request a *feature* or report a *bug*?** bug **What is the current behavior?** Throwing a resolved promise inside an app mounted with 'React.unstable_createRoot().render()' triggers the suspense fallback render, defocusing inputs in the app. In contrast, throwing an immediately resolved promise inside the same app rendered with 'ReactDOM.render()' does not trigger the suspense fallback render. Here's a small reproduction showing the different behaviors: ```javascript import React from "react"; import ReactDOM from "react-dom"; let cache = {}; function MyApp() { let [text, setText] = React.useState("edit this"); </pre>	Other	Insufficient Sample Data	Input Insufficient

16319	0	Other	16319	componentDidUpdate not triggered on changes to context	NONE	<p>**Do you want to request a *feature* or report a *bug*?** Bug.</p> <p>**What is the current behavior?** Components using a context update their content upon changes to the context, but `render` and `componentDidUpdate` are not invoked even though the components' content changes.</p> <p>[Here's a JSFiddle example](https://jsfiddle.net/0ewuj8L4/). Note how the component does update (the display on screen changes), but the "render" messages are only logged once (to the console), while the "update" messages are never logged.</p> <p>**What is the expected behavior?** I guess I understand why this is happening - the components which use `Context.Consumer` don't really re-render or get updated when the context changes; only the `Context.Consumer` component does. It would still be appreciated to at least make `componentDidUpdate` get invoked somehow (automatically).</p> <p>Regardless, this behavior should certainly be documented as it is quite unclear, unintuitive and not so easy to detect.</p> <p>**Which versions of React, and which browser / OS are affected by this issue? Did this work in previous versions of React?** Latest React I guess? I'm running on Windows 10.0.17134.799 and Chrome 75.0.3770.142, but I believe it should replicate on other environments as well.</p>	Support	Insufficient Sample Data	Input Insufficient
16358	0	Other	16358	React state values is shared between two components!!!	NONE	<p>**Bug**</p> <p>I have two components: 'ComponentA'</p> <pre> '''js class ComponentA extends Component { constructor(props) { super(props); this.nextCardSet = this.nextCardSet.bind(this); this.prevCardSet = this.prevCardSet.bind(this); } async prevCardSet() { const currPage = this.state.currPage - 1; const data = this.state.dataStore[currPage - 1]; await this.setState({ currPage, data }); } async nextCardSet() { const currPage = this.state.currPage + 1; const data = this.state.dataStore[currPage - 1]; await this.setState({ currPage, </pre>	Support	Insufficient Sample Data	Input Insufficient

16370	1	Support	16370	<p>useEffect does not get executed again after 16.8.0-alpha1</p>	<p>**Do you want to request a *feature* or report a *bug*?** It is a bug</p> <p>**What is the current behavior?** As seen in this [codepen](https://codesandbox.io/s/react-hooks-playground-k8hxy), the useEffect gets updated every 500ms as expected and mentioned in the docs:</p> <p>> Does useEffect run after every render? Yes! By default, it runs both after the first render and after every update.</p> <p>But if the react version is updated to a newer version, the timeout is called, the setCounter is called with 1 again (after the second run) which leads to a rerender of the component, but the effect is not executed as expected (2 is not called again and there is no loop). As mentioned in the docs, use Effect without a second parameter should create a loop by calling the setTimeout again after setCount. But this is not the case for versions above 16.8.0-alpha1. Is this intended or not?</p> <p>**If the current behavior is a bug, please provide the steps to reproduce and if possible a minimal demo of the problem. Your bug will get fixed much faster if we can run your code and it doesn't have dependencies other than React. Paste the link to your JSFiddle (https://jsfiddle.net/Luktwrmd/) or CodeSandbox (https://codesandbox.io/s/new) example below:** This [codepen](https://codesandbox.io/s/react-hooks-playground-k8hxy) works by creating a infinity loop. But updating the react version here ![image](https://user-images.githubusercontent.com/17567991/62890380-aa5a1680-bd43-11e9-8f6c-0e026510365a.png)</p> <p>will break the functionality and will not loop again because the useEffect is not executed if the previousState === currentState. But the component gets rerendered anyway.</p>	Other	Insufficient Sample Data	Input Insufficient
16419	0	Other	16419	<p>DevTools: react-devtools-tutorial.now.sh > editing-props-and-state have a bad state</p>	<p>I don't know who's making the bug, so reporting here. I'll move to correct repo if someone helps me debug it.</p> <p>**Do you want to request a *feature* or report a *bug*?** Bug</p> <p>**What is the current behavior?** Please watch below screencast: https://drive.google.com/file/d/1KMP44qsZ4y3MwrLLDdnOzPZ8z5mMEIFP/view</p> <p>1. Goto https://react-devtools-tutorial.now.sh/editing-props-and-state 2. Change the last ListItem prop to isComplete from 'false' to 'true'. 3. Click the checkbox in the view to change the state again from 'true' to 'false'.</p> <p>**What is the expected behavior?** It should just change the state of that ListItem. Instead, it's adding 3 more in the list with duplicate keys.</p> <p>**Which versions of React, and which browser / OS are affected by this issue? Did this work in previous versions of React?** Latest React. Mac, Chrome Version 75.0.3770.142 (Official Build) (64-bit)</p>	Bug	Input Insufficient	Insufficient Sample Data, imbalanced Data

16604	0	Other	16604	How should we set up apps for HMR now that Fast Refresh replaces react-hot-loader?	NONE	<p>Dan Abramov mentioned that Devtools v4 will be making 'react-hot-loader' obsolete: https://twitter.com/dan_abramov/status/1144715740983046144?s=20</p> <p>> **Me:** > I have this hook: > ```require("react-reconciler")(hostConfig).injectIntoDevTools(opts);``` > But HMR has always worked completely without it. Is this now a new requirement?</p> <p>> **Dan:** > Yes, that's what the new mechanism uses. The new mechanism doesn't need "react-hot-loader" so by the time you update, you'd want to remove that package. (It's pretty invasive)</p> <p>I can't see any mention of HMR in the Devtools documentation, however; now that 'react-hot-loader' has become obsolete (and with it, the 'require("react-hot-loader/root").hot' method), how should we set up apps for HMR in:</p> <ul style="list-style-type: none"> * React DOM apps * React Native apps * React custom renderer apps <p>I'd be particularly interested in a migration guide specifically for anyone who's already set up HMR via 'react-hot-loader'.</p> <p>Also, for HMR, does it matter whether we're using the standalone Devtools or the browser-extension Devtools?</p>	Support	Insufficient Sample Data	Input Insufficient
16957	0	Other	16957	Webkit inline styles dissapears in Firefox	NONE	<p>**Do you want to request a *feature* or report a *bug*?*: *bug*</p> <p>**What is the current behavior?** Missing '-webkit-print-color-adjust: exact;' in Firefox</p> <p>**Demo**: 1. Run https://stackblitz.com/edit/react-skxixb in Firefox and Chrome 2. Check CSS for '.header'</p> <p>**What is the expected behavior?** '-webkit-print-color-adjust: exact;' should be placed in all browsers</p> <p>**Which versions of React, and which browser / OS are affected by this issue? Did this work in previous versions of React?** React 16.9.0 and 16.10.1 Windows 10 Firefox 69.0.1</p>	Bug	Input Insufficient	Insufficient Sample Data, imbalanced Data

17182	0	Other	17182	React_ createContext is not a function	NONE	<p>Hi so i have been trying to implement react context api into my project, After follow the steps in the guide: https://developerhandbook.com/react/build-a-complete-property-listings-page-with-react/</p> <p>I ended up getting an error when i tried to display some of the information.</p> <p>This is the error i got in my console log:</p> <pre> ... OrderListProvider.js:6 Uncaught TypeError: __WEBPACK_IMPORTED_MODULE_0_react__._createContext is not a function at Object../src/context/OrderListProvider.js (OrderListProvider.js:6) at __webpack_require__ (bootstrap 39db4eed0e38b5656c68:678) at fn (bootstrap 39db4eed0e38b5656c68:88) at Object../src/components/Home.js (FetchData.js:3) at __webpack_require__ (bootstrap 39db4eed0e38b5656c68:678) at fn (bootstrap 39db4eed0e38b5656c68:88) at Object../src/App.js (fetch.js:461) at __webpack_require__ (bootstrap 39db4eed0e38b5656c68:678) at fn (bootstrap 39db4eed0e38b5656c68:88) at Object../src/index.js (index.css?f255:26) at __webpack_require__ (bootstrap 39db4eed0e38b5656c68:678) at fn (bootstrap 39db4eed0e38b5656c68:88) at Object.0 (registerServiceWorker.js:108) at __webpack_require__ (bootstrap 39db4eed0e38b5656c68:678) at bootstrap 39db4eed0e38b5656c68:724 at bootstrap 39db4eed0e38b5656c68:724 ... </pre>	Support	Insufficient Sample Data	Input Insufficient
17332	0	Other	17332	useTransition: After startTransition, it does not react to passed props changes	NONE	<pre> <!-- Note: if the issue is about documentation or the website, please file it at: https://github.com/reactjs/reactjs.org/issues/new --> **Do you want to request a *feature* or report a *bug*?** Probably a bug **What is the current behavior?** After firing startTransition, "current" component stops reacting to passed props changes while reacting to local state changes. **If the current behavior is a bug, please provide the steps to reproduce and if possible a minimal demo of the problem. Your bug will get fixed much faster if we can run your code and it doesn't have dependencies other than React. Paste the link to your JSFiddle (https://jsfiddle.net/Luktwrdf) or CodeSandbox (https://codesandbox.io/s/new) example below:** In the following CodeSandbox, `count` is counting up in the parent component using `setInterval` but if we click "CLICK ME", it suddenly stops updating. https://codesandbox.io/s/usetransition-stop-reacting-passed-props-updates-p9k1b **What is the expected behavior?** When passed props change, it should show the latest value of it where possible **Which versions of React, and which browser / OS are affected by this issue? Did this work in previous versions of React?**) Experimental build (0.0.0-experimental-5faf377df) </pre>	Bug	Input Insufficient	Insufficient Sample Data, imbalanced Data
17431	0	Other	17431	[react-test-renderer] "Warning: Each child in a list should have a unique "key" prop."	NONE	<p>Hello!</p> <p>react-test-renderer works just fine, but I'm getting the following Warning on all of my Tests:</p> <pre> > Warning: Each child in a list should have a unique "key" prop. Has this been addressed? It isn't something related to my code or Tests. Many thanks! </pre>	Support	Input Insufficient	Insufficient Sample Data

17517	0	Other	17517	Stale values for useState inside callback functions	NONE	<p><!-- Note: if the issue is about documentation or the website, please file it at: https://github.com/reactjs/reactjs.org/issues/new --></p> <p>**Do you want to request a *feature* or report a *bug*?** potential bug</p> <p>**What is the current behavior?** When calling 'setState' from 'useState', the 'state' value is stale from inside a function. The value is correct inside the body of the component, but not inside the function itself.</p> <p>**If the current behavior is a bug, please provide the steps to reproduce and if possible a minimal demo of the problem. Your bug will get fixed much faster if we can run your code and it doesn't have dependencies other than React. Paste the link to your JSFiddle (https://jsfiddle.net/Luktwrmd/) or CodeSandbox (https://codesandbox.io/s/new) example below:** Begin by typing 'a@g' which should autocomplete to 'a@gmail.com'. You will see the 'expected' and 'actual' values for the input below each. https://codesandbox.io/s/useemailautocomplete-material-ui-04423</p> <p>**What is the expected behavior?** It should have the correct value for 'email' inside the 'handleChange' function.</p> <p>**Which versions of React, and which browser / OS are affected by this issue? Did this work in previous versions of React?* React: 'v16.12.0' Browser: chrome</p>	Support	Insufficient Sample Data	Input Insufficient
17677	1	Support	17677	How can i create a dynamic ENum	NONE	<p>This request asking about existing features supported by ReactJS. I have an enum in use on several reactJS pages(200 implementations across 32 code files). Now the requirement is to make it dynamic. I am not sure how can i achieve it in ReactJS+REDUX implementation.</p> <p>I am working on the latest version of ReactJS</p>	Other	Input Insufficient	Insufficient Sample Data
17911	1	Support	17911	Bug: startTransition suspends immediately when useLayoutEffect is present	NONE	<p><!-- Please provide a clear and concise description of what the bug is. Include screenshots if needed. Please test using the latest version of the relevant React packages to make sure your issue has not already been fixed. --></p> <p>React version: 0.0.0-experimental-f42431abe</p> <p>Please note that I do realize my repro steps are poor at best. I'm *not* filing this issue in hopes of support; I'm only filing this issue to provide one more datum point to help diagnose what I believe to be a bug, which I'm assuming you'll see more reports of.</p> <p>## Steps To Reproduce</p> <p>tl;dr - there are some circumstances when a thrown promise inside a hook causes an immediate suspense, instead of respecting the startTransition it's inside of.</p> <p>startTransition for me is always, in this case, called outside of the normal React handlers. In this case history.listen</p> <p>https://github.com/arackaf/booklist/blob/special/suspense-blog/react/modules/books/booksSearchState.ts#L75</p> <p>This is my Suspense-enabled hook that's called as a result of the state update inside the code above</p> <p>https://github.com/arackaf/micro-graphql-react/blob/feature/suspense/src/useQuery.js#L22</p> <p>the Promise throwing happens here</p>	Bug	Input Insufficient	Insufficient Sample Data, imbalanced Data

17927	1	Support	17927	Bug: Autocomplete not working for controlled input	NONE	<p>React version: 16.12.0</p> <p>## Steps To Reproduce</p> <ol style="list-style-type: none"> 1. Have a form with controlled input (i.e. value is set through 'onChange' handler) 1. Type 'cheese' into the input 1. Submit the form 1. Reload the page 1. Focus the input, type 'c' 1. 'cheese' is *not* suggested <p>https://codesandbox.io/s/ancient-carrying-oqgt6</p> <p>It works when an uncontrolled input is used (i.e. value is not set by react 'onChange' handler)</p> <ol style="list-style-type: none"> 1. Have a form with uncontrolled input 1. Type 'cheese' into the input 1. Submit the form 1. Reload the page 1. Focus the input, type 'c' 1. 'cheese' is being suggested <p>https://codesandbox.io/s/naughty-dijkstra-p3n42</p> <p>It's not about autofilling address data or passwords, but data previously filled in by the user. We noticed this issue in Chrome, Firefox and Safari. Even though we could not get any autocompletion to work in Safari, even without React. (We could in Chrome and Firefox)</p> <p>Thanks!</p>	Bug	Input Insufficient	Insufficient Sample Data, imbalanced Data
17953	0	Other	17953	Bug: useReducer runs the queued updates with new props	NONE	<p><!-- Please provide a clear and concise description of what the bug is. Include screenshots if needed. Please test using the latest version of the relevant React packages to make sure your issue has not already been fixed. --></p> <p>React version: 16.8.0</p> <p>## Steps To Reproduce</p> <p><!-- Your bug will get fixed much faster if we can run your code and it doesn't have dependencies other than React. Issues without reproduction steps or code examples may be immediately closed as not actionable. --></p> <p>Link to code example: https://codesandbox.io/s/usereducer-wfcmq Link to codesandbox.</p> <p><!-- Please provide a CodeSandbox (https://codesandbox.io/s/new), a link to a repository on GitHub, or provide a minimal code example that reproduces the problem. You may provide a screenshot of the application if you think it is relevant to your bug report. Here are some tips for providing a minimal example: https://stackoverflow.com/help/mcve. --></p>	Bug	Input Insufficient	

18183	3	Feature	18183	useMutableSource and hydration	COLLABORATOR	<p>Follow up to PR #18000 and RFC https://github.com/reactjs/rfcs/pull/147</p> <p>The new `useMutableSource` hook will need at least one additional API to be able to support server rendering and hydration. This API would likely be on the React root (the object returned by `createRoot`) and would enable mutable sources to have their versions eagerly captured before hydration begins.</p> <p>##### How is the version used?</p> <p>Currently a work-in-progress version of each mutable source is stored on the source itself. This version enables us to [avoid tearing before a source has been subscribed to](https://github.com/bvaughn/rfcs/blob/useMutableSource/text/0000-use-mutable-source.md#reading-from-a-source-before-subscribing). This version number is lazily populated (the first time a source is read during a given render).</p> <p>##### How will the version be used during hydration?</p> <p>In the case of server rendering, this version will need to be eagerly populated for every source so that we can detect tearing between the version of the source used for the server respond, and the version we will eventually read from while hydrating on the client.</p> <p>##### How will this work?</p> <p>One way to do this would be to store an array on each React root of mutable source and version number pairs. Each time we start (or resume) hydration, we can iterate through this array and use it to initialize the work-in-progress version for each source.</p> <p>Once all outstanding hydration work is finished, we can clear the array. (Although this will require an additional</p>	Bug	Imbalanced Data	Insufficient Sample Data
18205	1	Support	18205	Bug: backend.js Uncaught TypeError: Cannot read property 'sub' of undefined	NONE	<p>''' backend.js:32 Uncaught TypeError: Cannot read property 'sub' of undefined at g (backend.js:32) at e (backend.js:8) '''</p> <p>It happens when a server returns a string without HTML. I suppose React dev tools expect DOM.</p> <p>!Screenshot 2020-03-03 at 21 43 42](https://user-images.githubusercontent.com/66249/75817818-18c1fd00-5d98-11ea-89f5-547e97c44632.png)</p>	Bug	Input Insufficient	Insufficient Sample Data, imbalanced Data

18426	1	Support	18426	COLLABORATOR	<p><!-- Please provide a clear and concise description of what the bug is. Include screenshots if needed. Please test using the latest version of the relevant React packages to make sure your issue has not already been fixed. --></p> <p>React version: 16.3-16.13</p> <p>## Steps To Reproduce</p> <ol style="list-style-type: none"> 1. Render function component with side-effects and without hooks in StrictMode 2. Component only renders once <p>Link to code example: https://codesandbox.io/s/strictmode-w-and-wo-hooks-vgxvh</p> <p>## The current behavior</p> <p>StrictMode only renders function components with hooks twice following https://github.com/facebook/react/issues/15074#issuecomment-471197572</p> <p>## The expected behavior</p> <p>I think making [the distinction between components with and without hooks causes more confusion than it helps](https://github.com/mui-org/material-ui/issues/20313). Especially since the docs do not mention this. I</p>	Feature	Insufficient Sample Data	Imbalanced Data
18512	1	Support	18512	CONTRIBUTOR	<p>Bug: dev tools development script is running production build</p> <p><!-- Please provide a clear and concise description of what the bug is. Include screenshots if needed. Please test using the latest version of the relevant React packages to make sure your issue has not already been fixed. --></p> <p>React Dev Tools version: 4.6.0</p> <p>I'm not sure there is a problem with 'package.json' within 'react-devtools-extensions' package, or it's just my misunderstanding how it works, but script 'build:dev' create production build. Command above runs a each script (per browser) in which 'NODE_ENV' is set to 'production'.</p> <p>## Steps To Reproduce</p> <ol style="list-style-type: none"> 1. Just run 'yarn build:dev' in 'react-devtools-extensions' 2. It creates minified version of files. <p>https://github.com/facebook/react/blob/master/packages/react-devtools-extensions/package.json</p> <pre> '''javascript "scripts": { "build": "cross-env NODE_ENV=production yarn run build:chrome && yarn run build:firefox && yarn run build:edge", "build:dev": "cross-env NODE_ENV=development yarn run build:chrome && yarn run build:firefox && yarn run build:edge", "build:chrome": "cross-env NODE_ENV=production node ./chrome/build", </pre>	Bug	Imbalanced Data	Insufficient Sample Data, Input Insufficient

18669	0	Other	18669	MEMBER	<p>Thereâ€™s some things we donâ€™t have a sufficient coverage of. Currently we catch them from production product bugs but this is not sustainable. Weâ€™re hoping some refactors will drastically simplify the model â€” but nevertheless we should invest in better fuzz test coverage. Thatâ€™s how we caught similar bugs before at an earlier stage.</p> <p>One thing weâ€™re lacking coverage for is what happens to Suspense boundaries as updates are dispatched at different priorities in different order, and what happens when weâ€™ve had to yield. We need to verify that Suspense always â€œwakes upâ€ when Promises are resolved and thereâ€™s nothing to be suspended on. We also need to test this in combination with render phase updates.</p> <p>Hereâ€™s examples of bugs that I want a fuzzer to catch: https://github.com/facebook/react/issues/18657 https://github.com/facebook/react/issues/18020 https://github.com/facebook/react/issues/18486 https://github.com/facebook/react/issues/18357 https://github.com/facebook/react/issues/18353 https://github.com/facebook/react/issues/18644 https://github.com/facebook/react/pull/18412.</p> <p>@dubzzz I believe you were interested in this? This would take some effort but would be a major contribution.</p>	Feature	Insufficient Sample Data	Imbalanced Data
18821	1	Support	18821	NONE	<p>Bug: ARIA Attribute Reflection</p> <p>React version: 16.13.1</p> <p>## Steps To Reproduce</p> <ol style="list-style-type: none"> 1. Implement the [gov.uk "breadcrumbs" component](https://design-system.service.gov.uk/components/breadcrumbs/) in React. 2. Use the [ARIA 1.2](https://www.w3.org/TR/wai-aria-1.2/#idl-interface) 'ariaCurrent' property, as available in Edge 81, Chrome 81, and Safari 13. 3. See warning: <pre> Warning: Invalid ARIA attribute `ariaCurrent`. Did you mean `aria-current`? </pre> <p>Link to code example: [https://codesandbox.io/s/bold-glitter-lpfpq](https://codesandbox.io/s/bold-glitter-lpfpq?file=/src/App.js)</p> <pre> ```jsx function Breadcrumbs() { return (Home </pre>	Other	Insufficient Sample Data	Input Insufficient

18831	0	Other	18831	<p>Error: "Commit tree does not contain fiber 256. This is a bug in React DevTools."</p> <p>Describe what you were doing when the bug occurred: 1. Profiling a slow component In a component that rendered 5000 pre tags with single lines of text in them, that has an unrelated controlled text box is the same component that was typed into while profiling. App hung a while and, when it rendered again the error was in the profiler.</p> <p>----- Please do not remove the text below this line -----</p> <p>DevTools version: 4.6.0-6cceaeb67</p> <p>Call stack: at j (chrome-extension://fmkadmappgofadopljbjfkapdkoiieni/build/main.js:40:162825) at N (chrome-extension://fmkadmappgofadopljbjfkapdkoiieni/build/main.js:40:161628) at e.getCommitTree (chrome-extension://fmkadmappgofadopljbjfkapdkoiieni/build/main.js:40:164582) at ec (chrome-extension://fmkadmappgofadopljbjfkapdkoiieni/build/main.js:40:339280) at ci (chrome-extension://fmkadmappgofadopljbjfkapdkoiieni/build/main.js:32:59620) at Ll (chrome-extension://fmkadmappgofadopljbjfkapdkoiieni/build/main.js:32:109960) at qc (chrome-extension://fmkadmappgofadopljbjfkapdkoiieni/build/main.js:32:102381) at Hc (chrome-extension://fmkadmappgofadopljbjfkapdkoiieni/build/main.js:32:102306) at Vc (chrome-extension://fmkadmappgofadopljbjfkapdkoiieni/build/main.js:32:102171) at Tc (chrome-extension://fmkadmappgofadopljbjfkapdkoiieni/build/main.js:32:98781)</p> <p>Component stack: in ec in div in div in div</p>	Bug	Imbalanced Data	Insufficient Sample Data, Input Insufficient
19006	0	Other	19006	<p>Bug: ReactPartialRenderer function createOpenTagMarkup calls isCustomComponent for every property unnecessarily</p> <p>NONE</p> <p>This is a performance issue. Function isCustomComponent is called multiple times for each property of the same component inside of createOpenTagMarkup function (ReactPartialRenderer.js).</p>	Bug	Input Insufficient	Insufficient Sample Data, imbalanced Data
19151	1	Support	19151	<p>React Developer Tool Is not working in my localhost. It was working fine before.</p> <p>Hi Team</p> <p>I am not able to use the react developer tool extension on my chrome browser, Which was working well before. I have tried by removing and re adding the extension again. Still the problem persists.</p> <p>I have attached a screenshot of the tool icon on browser. I am very new to react Environment. Appreciate your help here.</p> <p>Thank you.</p> <p>![image](https://user-images.githubusercontent.com/28383863/84981630-8f6de500-b152-11ea-909f-f386f226d7ee.png)</p> <p>NONE</p>	Bug	Input Insufficient	Insufficient Sample Data, imbalanced Data
19229	2	Bug	19229	<p>CI missing failing tests?</p> <p>COLLABORATOR</p> <p>PR #19222 recently landed with no CI failures, but `ReactDOMServerSelectiveHydration` was definitely broken as a result of this PR (and was subsequently fixed in #19227). This is just a reminder for one of us to dig into why CI didn't catch this failure.</p>	Other	Input Insufficient	Insufficient Sample Data
19662	0	Other	19662	<p>Add a toggle for Boolean props in DevTools</p> <p>MEMBER</p> <p>We previously had a feature where Boolean props would show a checkbox to the left of them in the DevTools pane. It was removed when the JSON editor was added, but I think we should add it back. It should work like this:</p> <p>1. If the value is a boolean, the checkbox should show up to the left of `true` / `false` value 2. If it's no longer a boolean (e.g. gets edited manually), the checkbox disappears</p>	Feature	Insufficient Sample Data	Imbalanced Data

19747	0	Other	19747	<p>Bug: Using opacity as a percentage value in a css file will become 1% in the production build.</p>	NONE	<p>Using opacity as a percentage value in a css file will become 1% in the production build version, even though in the localhost website it will appear correctly. The way around this is to use a decimal value (ex. 0.95) for opacity.</p> <p>React version: 16.13.1</p> <p>## Steps To Reproduce</p> <ol style="list-style-type: none"> 1. Set a css class to have a percentage opacity value (ex. 95%) 2. run npm run-scrips build 3. build version will have incorrect percentage (1%) <p>code example:</p> <pre>.App { font-family: sans-serif; text-align: center; opacity: 50%; }</pre> <p>## The current behavior Opacity will change from any percentage value to 1% in app build version.</p> <p>## The expected behavior Opacity will stay the same percentage in build version.</p>	Bug	Imbalanced Data	Insufficient Sample Data, Input Insufficient
19861	1	Support	19861	<p>Error: "child is undefined"</p>	NONE	<p>Hi! I was developing a next.js app, then I open the dev tools to see the state of some component, and the react dev tools give this error, after a page refresh all when back to normal, I don't know if this is a bug caused by me or is something related to the dev tools itself, so I prefer to report it, if this my fault I'm very sorry for the trouble!</p> <p>-----</p> <p>Please do not remove the text below this line</p> <p>-----</p> <p>DevTools version: 4.8.2-fed4ae024</p> <p>Component stack: List@moz-extension://e8970055-c8eb-4403-952e-40ddc0e5c62a/build/main.js:20924:30 div AutoSizer@moz-extension://e8970055-c8eb-4403-952e-40ddc0e5c62a/build/main.js:2786:19 div div Tree_Tree@moz-extension://e8970055-c8eb-4403-952e-40ddc0e5c62a/build/main.js:26368:45 div div InspectedElementContextController@moz-extension://e8970055-c8eb-4403-952e-40ddc0e5c62a/build/main.js:26848:18 OwnersListContextController@moz-extension://e8970055-c8eb-4403-952e-40ddc0e5c62a/build/main.js:25520:18 SettingsModalContextController@moz-extension://e8970055-c8eb-4403-952e-40ddc0e5c62a/build/main.js:26139:18 Components_Components@moz-extension://e8970055-c8eb-4403-952e-40ddc0e5c62a/build/main.js:30926:50 ErrorBoundary@moz-extension://e8970055-c8eb-4403-952e-40ddc0e5c62a/build/main.js:27172:33 PortaledContent@moz-extension://e8970055-c8eb-4403-952e-40ddc0e5c62a/build/main.js:27303:27</p>	Other	Input Insufficient	Insufficient Sample Data

20022	0	Other	20022	Bug: useRef forget '()	CONTRIBUTOR	<p><!-- Please provide a clear and concise description of what the bug is. Include screenshots if needed. Please test using the latest version of the relevant React packages to make sure your issue has not already been fixed.</p> <p>--></p> <p>React version:</p> <p>## Steps To Reproduce</p> <ol style="list-style-type: none">1.2. <p><!-- Your bug will get fixed much faster if we can run your code and it doesn't have dependencies other than React. Issues without reproduction steps or code examples may be immediately closed as not actionable.</p> <p>--></p> <p>Link to code example: https://codesandbox.io/s/dazzling-heyrovsky-18u9w?file=/src/App.js</p> <p><!-- Please provide a CodeSandbox (https://codesandbox.io/s/new), a link to a repository on GitHub, or provide a minimal code example that reproduces the problem. You may provide a screenshot of the application if you think it is relevant to your bug report. Here are some tips for providing a minimal example: https://stackoverflow.com/help/mcve.</p>	Support	Input Insufficient	Insufficient Sample Data
20070	1	Support	20070	Bug: Property expression of JSXExpressionContainer expected node to be of a type ["Expression", "JSXEmptyExpression"] but instead got undefined	NONE	<p>React version: 17.0.0 Next.js version: 9.5.5 TypeScript version: 4.0.3</p> <p>## The current behavior</p> <p>In a Next.js project using styled-jsx I get the following error across different components that create a `<style jsx>` tag to generate CSS selectors within template literals (or import such styles from a separate file):</p> <pre>...</pre> <p>Property expression of JSXExpressionContainer expected node to be of a type ["Expression", "JSXEmptyExpression"] but instead got undefined</p> <pre>...</pre> <p>The error appears as soon as I update `react` and `react-dom` to 17.0.0, all other dependencies remain at the version they have been beforehand.</p> <p>## The expected behavior</p> <p>It renders the component as expected without any errors.</p>	Other	Insufficient Sample Data	Input Insufficient
20334	0	Other	20334	Bug: Strange lines in the page	NONE	<p>When chrome devtools is opened, some strange lines appear in the page and disappear after disable&enable the react extension.</p> <p>I'm wondering if there are some bugs on react extension.</p> <p>Win10x64 Chrome 87.0.4280.66 [Chrome issue 83155766](https://support.google.com/chrome/thread/83155766) </p>	Support	Input Insufficient	Insufficient Sample Data

20339	1	Support	20339	Improved "memory leak" warning	NONE	<p>Dear React Maintainers,</p> <p>My proposal to improve the ["memory leak" warning](https://github.com/facebook/react/blob/e6a0f276307fcb2f1c5bc41d630c5e4c9e95a037/packages/react-reconciler/src/ReactFiberWorkLoop.new.js#L3096) has quite a long background, so let me first thank you for keeping React alive and well. You are all fabulous ðŸŽ‰</p> <p>### TL;DR</p> <p>A Promise is not cancellable, so there is 90% chance that no memory leaks will be fixed when a user applies a "solution" to this warning from the internet. This warning encourages a pit of failure (more complex code without removing actual memory leaks) and I argue the detection of memory leaks should be changed.</p> <p>Table of contents: The good Intended Solutions - The bad - The uncanny - Proposal</p> <p># The good </p> <p>> Can't perform a React state update on an unmounted component. This is a no-op, but it indicates a memory leak in your application. To fix, cancel all subscriptions and asynchronous tasks in a useEffect cleanup function.</p> <p>There were, obviously, good reasons to introduce this warning in the first place. React can't detect memory leaks directly, so this was the next best thing to detect forgotten imperatively attached DOM event handlers or uncancelled Web APIs. The warning itself does not list any examples, so let me illustrate with [my own bad example](https://codesandbox.io/s/immutable-sun-fo3nt?file=/src/App.js):</p> <pre> '''js </pre>	Other	Insufficient Sample Data	Input Insufficient
20450	0	Other	20450	Bug: Trailing Comma in Arrays and Dict	NONE	<p>The learn react tic tac toe game tutorial shows a trailing comma at the end of dictionary values. This is known to cause issues with Internet Explorer. Hoping you can change that.</p> <p>https://stackoverflow.com/questions/5139205/javascript-can-a-comma-occur-after-the-last-set-of-values-in-an-array</p> <p>React version: LTS</p> <p>## Steps To Reproduce</p> <p>1. Run tic tac toe tutorial code</p> <pre> <!-- Your bug will get fixed much faster if we can run your code and it doesn't have dependencies other than React. Issues without reproduction steps or code examples may be immediately closed as not actionable. --> </pre> <p>Link to code example:</p> <pre> <!-- Please provide a CodeSandbox (https://codesandbox.io/s/new), a link to a repository on GitHub, or provide a minimal code example that reproduces the problem. You may provide a screenshot of the application if you think it is relevant to your bug report. Here are some tips for providing a minimal example: https://stackoverflow.com/help/mcve. --> </pre>	Support	Input Insufficient	

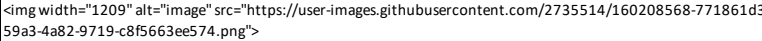
20590	1	Support	20590	Bug: react-test-renderer provides no equivalent to attachTo option of Enzyme, mandatory to snapshot Leaflet and other	NONE	<!-- Please provide a clear and concise description of what the bug is. Include screenshots if needed. Please test using the latest version of the relevant React packages to make sure your issue has not already been fixed. --> React version: 16.13.1 ## Steps To Reproduce 1. In a project with React, and react-leaflet@2.7.0 2. ```ts import React from "react"; import Renderer from "react-test-renderer"; import { Map, TileLayer } from "react-leaflet"; let component = Renderer.create(<Map> <TileLayer attribution="© OpenStreetMap contributors' url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png" /> </Map>); ```	Other	Insufficient Sample Data	Input Insufficient
20841	0	Other	20841	Mixin rewrite with React and Higher-order-functions	NONE	@gaearon How would you re-write the following mixins? https://github.com/gerobit/ivis-core/blob/a51154c8937d4d392f20303caeb8b450ecf9189a/client/src/lib/decorator-helpers.js	Support	Input Insufficient	
21098	0	Other	21098	Bug: aspectRatio not being applied via style	NONE	React version: 16.14.0, 17.0.2 ## Steps To Reproduce 1. Apply aspectRatio via `style` - no aspectRatio applied 2. Add via className - aspectRatio applied Needs Chrome 79 or above: Link to code example: https://codesandbox.io/s/react-bug-not-applying-aspect-ratio-via-style-w2pqh?file=/src/App.js ## The current behavior No aspectRatio rule applied: https://codesandbox.io/s/react-bug-not-applying-aspect-ratio-via-style-w2pqh?file=/src/App.js ## The expected behavior aspectRatio applied, here I'm adding it via a `useLayoutEffect`: https://codesandbox.io/s/currying-hill-i09qn?file=/src/App.js	Support	Input Insufficient	

21191	0	Other	21191	useCallback memoization	NONE	<p>I want to continue a discussion about the 'useCallback' that started on twitter (https://twitter.com/ramon_fritsch/status/1379569928454303749) here, following @gaearon's suggestion.</p> <p>It's about 'useCallback' returning a new reference when dependencies change. I always wondered if that's what we usually want.</p> <p>Take this example: https://codesandbox.io/s/sipxs</p> <p>I understand 'useCallback' is intended to use on event handlers and regular callback props, thus we need to make it as flexible as possible. But the case for event handlers, I'd suggest to have a new hook 'useEventCallback' that would memoize the callback function forever and never return a new reference, helping in not causing a re-render down the chain.</p> <p>The folks from Formium solved this with https://github.com/formium/formik/blob/2d613c11a67b1c1f5189e21b8d61a9dd8a2d0a2e/packages/formik/src/Formik.tsx#L1193-L1205</p> <p>Isn't it important enough to become a core hook in the React ecosystem? Or at least have a clear statement in the documentation about this behavior and best practices for event handlers?</p>	Support	Insufficient Sample Data	Input Insufficient
21371	1	Support	21371	Bug: calling calling 'setState' twice inside 'useEffect' creates extra function calls	NONE	<pre><!-- Please provide a clear and concise description of what the bug is. Include screenshots if needed. Please test using the latest version of the relevant React packages to make sure your issue has not already been fixed. --> React version: 17.0.2 ## Steps To Reproduce 1. Create a function component 2. Call 2 'setState' functions inside a 'useEffect' hook. <!-- Your bug will get fixed much faster if we can run your code and it doesn't have dependencies other than React. Issues without reproduction steps or code examples may be immediately closed as not actionable. --> ```tsx import { FC, useEffect, useState } from 'react' const resolvedPromise = Promise.resolve() const Test: FC = () => { const [a, setA] = useState(2) const [b, setB] = useState(0)</pre>	Other	Insufficient Sample Data	Input Insufficient

21390	0	Other	21390	Bug: Cannot read property __SECRET_INTERNALS_DO_NO T_USE_OR_YOU_WILL_BE_FIRE D of undefined	NONE	<!-- Please provide a clear and concise description of what the bug is. Include screenshots if needed. Please test using the latest version of the relevant React packages to make sure your issue has not already been fixed. --> # I got this issue when using react in deno deploy, but i think it will still throw this error in nodejs ## Code: https://github.com/code913/react-ssr-deno ! [what the heck is this tbh] (https://user-images.githubusercontent.com/54856929/116580980-82ed1800-a931-11eb-8671-05cb0cde4dfc.png) React version: 17.0.2 react-dom, react version latest ## Steps To Reproduce 1. Import react-dom from the umd production or development url (haven't tested others): `https://cdnjs.cloudflare.com/ajax/libs/react-dom/17.0.2/umd/react-dom.development.min.js` 2. Try to hydrate some data from react-dom/server	Support	Input Insufficient	
21644	0	Other	21644	Request for Package Release	NONE	I'm sorry if this is the wrong place to file this, but I wasn't sure how else to get ahold of the React team. The #21392 by @bvaughn fixes an issue that was causing MobX components to display components in stack traces as `wrappedComponent`, making it very hard to debug issues. This PR was merged back in April, but there hasn't been a React release since March. Is there any chance you guys would be willing to release React 17.0.3 with this fix? Thanks!	Support	Input Insufficient	Insufficient Sample Data
21989	1	Support	21989	Bug: createPortal anywhere in the tree makes native events be ran too late	NONE	## Summary Native events added via `useEffect` are called too late (and with improper (new) state, rather than the state during listener attachment) if there's a `ReactDOM.createPortal` anywhere in the tree. First the effect is re-run and a new listener is attached, and only then the native event is called. React version: 17.0.2 Link to code example: [CodeSandbox] (https://codesandbox.io/s/late-star-1izm6) ## Reproduction * Anywhere in the tree is a `createPortal`. It can even be `createPortal(null, document.body)` * Somewhere, there is a component with `useState`. * It renders a div. The div has an `onClick` handler passed. * On top of that, `useEffect` is used to attach a `click` event to `window` * Both event handlers (both the native event attached to window, and the React event passed to `div`) use `setState`. ## The current behavior * The React handler is called. It calls `setState` * The component rerenders with the new state, and its effects are re-run. * Since the effect is re-run, a new native listener is attached (which is bound to the new state) * Only then, the new listener (with the NEW state) is called.	Other	Insufficient Sample Data	Input Insufficient

22048	1	Support	22048	Bug: Cannot set property 'memoizedState' of null with nested renderToStaticMarkup and hooks	NONE	<p>React version: 17.0.2</p> <p>## Steps To Reproduce</p> <ol style="list-style-type: none"> 1. Have a component which memoizes the result of a renderToStaticMarkup. 2. Use renderToStaticMarkup to render the component of 1. <p>Link to code example:</p> <pre> ... function nestedRender() { return renderToStaticMarkup(<p>This is the nested render result</p>); } function MyComponent() { const nestedRenderResult = useMemo(() => nestedRender(), []); // Removing useMemo will solve the issue return <p>The nested result is: {nestedRenderResult}</p>; } function wrapperRender() { return renderToStaticMarkup(<MyComponent />); } ... ` https://codesandbox.io/s/lingering-shadow-vyqyd </pre> <p>## The current behavior</p>	Other	Insufficient Sample Data	Input Insufficient
22115	0	Other	22115	DevTools: Better Bundle Names for Dynamically Imported Modules	CONTRIBUTOR	<p>In the DevTools extension, webpack currently uses an automatically assigned ID as the chunk name for dynamically imported modules (ie. 'parseHookNames' and associated code will get bundled into '6.js').</p> <p>We've tried adding 'chunkFilename: '[name].js"' to 'output' in 'webpack.config.js' and magic comments (ie. '/* webpackChunkName: "parseHookNames" */') to the dynamic import to fix, but neither works.</p>	Bug	Input Insufficient	Insufficient Sample Data, imbalanced Data
22133	1	Support	22133	refs for function components as first class citizens	NONE	<p>I'm looking everywhere, but I can't seem to find an official answer.</p> <p>It seems `ref`s are discouraged, and developers are required to use `forwardRef`, or pass ref as a separate prop (i.e. `forwardRef={ref}`).</p> <p>`ref` are very important when access to the DOM is required, usually when a measurement is needed.</p> <p>Example usages include <code>**tooltips**</code>, graph nodes, animations, and dom observers.</p> <p>For example, <code>[@tippyjs/react]</code>(https://www.npmjs.com/package/@tippyjs/react) assumes children receive a ref (since <code>'findDOMNode'</code> is deprecated), and will simply fail with most components.</p> <p>Another example is a <code>'ClickOutside'</code> hook, that watches click events and triggers callback which clicking on a dom element outside a specific component.</p> <p>Lastly, it really complicated component compositions.</p> <p>Composing components and ultimately forwarding all props down to the DOM allows us to use all components as first class citizens, so they all have the same abilities as a div. I can pass an event handlers, data attribute, tabIndex, etc, to any component, without having to alter it.</p> <pre> ````tsx interface UserMenuProps extends MenuProps { users: User[]; } function UserMenu({users, ...rest}: UserMenuProps) { // even if Menu supports ref forwarding, I need to forwardRef again just to keep the same behavior. return <Menu {...rest}> {props.users.map(x=> <UserItem user={x}/>)} </Menu> } </pre>	Other	Insufficient Sample Data	Input Insufficient

22301	1	Support	22301	found an empty rule set	CONTRIBUTOR	<p>this is the file path - packages/react-devtools-shared/src/devtools/views/Settings/SettingsShared.css</p> <p>going here i found a empty rule set called selector which had nothing inside it and it is not a good practice to keep empty ruleset .</p> <p>here is the PR link for the same - https://github.com/facebook/react/pull/22298</p> <p>hope to get a feedback soon!</p>	Other	Input Insufficient	Insufficient Sample Data
22758	1	Support	22758	Some one is changing the state variable from the react development tool in production build?	NONE		Other	Input Insufficient	Insufficient Sample Data
22997	0	Other	22997	Bug: Image source takes a string, but not a variable with the same string	NONE	<p>I'm using create-react-app</p> <p>In my App.js, (the standard file created automatically)</p> <pre> ... //This one works function Image(){ return } //This doesn't work function Image2(){ let src = "./image.png"; return } ... </pre> <p>Seems odd, since src has the same value as the string. Is it maybe a compiler error?</p> <p>Complete file would be like this</p> <pre> ... import React from 'react'; //This work function Image(){ </pre>	Support	Insufficient Sample Data	Input Insufficient
23026	0	Other	23026	eslint-plugin-react-hooks: exhaustive-deps nagging behaviour	NONE	<p>In some situations like the following, I can't find an elegant way to deal with exhaustive-deps error. I don't want to disable it just because of that maybe there is room to improve the linter rule here?</p> <pre> ...js const { firstName, lastName, middleName, email, ...(list goes on) } = someData; const cb = useCallback(() => { ... someFn({ firstName, lastName, middleName, ... }) //(basically a subset of someData with other properties from component context) }, [someData]) // => here I have to list all the variables I destructured from that object instead of just putting someData as dependency. When we assume that we use that destructured variables all over in the component context and they are numerous; dependency list grows long and maybe not so efficient. ... </pre>	Support	Insufficient Sample Data	Input Insufficient

23374	0	Other	23374	Bug: CustomError: Cannot find module react/jsx-runtime	NONE	<p><!-- Please provide a clear and concise description of what the bug is. Include screenshots if needed. Please test using the latest version of the relevant React packages to make sure your issue has not already been fixed. --></p> <p>React version: 17.0.2</p> <p>## Steps To Reproduce</p> <p>Not sure</p> <p>## The current behavior</p> <p>CustomError: Cannot find module react/jsx-runtime</p> <p>## The expected behavior</p> <p>No Error</p> <p>I switch from common js to esm & after done that I am getting this error. I added '.js' & here is my 'tsconfig'</p> <pre> ... "target": "es5", "module": "es2020", "lib": ["dom", "dom.iterable", "esnext"], "jsx": "react-jsx", "moduleResolution": "node", </pre>	Support	Input Insufficient	Insufficient Sample Data
24170	1	Support	24170	[React DevTools] Component Stacks for Timeline Profiler	CONTRIBUTOR	<p>In the Timeline Profiler, we currently denote each state update with a dot. If you hover on the state update, you get some information about it, such as which component caused the update, the lane the update was rendered at, and the time that the update happened. This is useful for unique components. However, for components (ex. library components) that are used in multiple places, just having the component name is less helpful.</p> <p>It would be most useful to get a stack of component owners (like in the rendered by section in the Components tab). However, we only have owner metadata in DEV mode, and it usually only makes sense to profile in production. The next best thing we can do, then, is to get all the parent components and create a stack out of that (ie the return path of the fiber rather than the owner path).</p>  <p>We want to add component stacks so that we also know which parent(s) caused the update. For this task, a potential solution is:</p> <ul style="list-style-type: none"> * [] When a state update happens, walk the fiber's return path and save all the component names and their source. (See 'markStateUpdateScheduled' for code pointers on where to put this) * [] After we're done profiling, process the stack so that we create a stack of return fibers * [] Pass the stack to the DevTools front end and add the component stacks to the UI when a user hovers over a state update 	Feature	Insufficient Sample Data	Imbalanced Data

24239	0	Other	24239	Bug: App is unresponsive with React 18	NONE	<p>Before upgrading to react 18 app was working well but after upgrading this is not responding well</p> <p>React version:18</p> <p>## Steps To Reproduce</p> <ol style="list-style-type: none"> 1. Open the application and wait for 2 sec. then see buttons are not clickable 2. After sometime we will see page unresponsive <p>Link to code example:</p> <p>Deployed URL : https://app-jitera.netlify.app/</p> <p>## The current behavior</p> <p>As you can see in the screenshot.</p> <p>## The expected behavior</p> <p>It should run without lag.</p>	Support	Input Insufficient	Insufficient Sample Data
24280	1	Support	24280	Bug: componentWillUnmount is called twice	NONE	<p>React version: 18.0.0</p> <p>## Steps To Reproduce</p> <p>'componentWillUnmount' is called twice upon toggling the rendered component. Even when StrictMode is disabled</p> <p>Link to code example: https://codesandbox.io/s/componentwillunmount-called-twice-hrpzy5?file=/src/App.js</p> <p>## The current behavior</p> <p>After upgrading to react 18 we've seen some different behavior in a conditionally rendered, lazy class component.</p> <p>In the provided code example the class component is rendered first. After the first toggle, the class component's componentWillUnmount is called twice.</p> <p>Subsequent toggle calls correctly lead to a single componentWillUnmount invocation.</p> <p>This does only seem to affect the class component when its rendered first. If the condition is changed to initially show the other function component the class component unmounts just fine</p> <p>## The expected behavior</p> <p>The class component's componentWillUnmount is only called once</p>	Bug	Imbalanced Data	Insufficient Sample Data, Input Insufficient

				Bug: Page renders twice when use "ReactDOM.createRoot" in the index.tsx in React 18		<!-- Please provide a clear and concise description of what the bug is. Include screenshots if needed. Please test using the latest version of the relevant React packages to make sure your issue has not already been fixed. --> React version:18.1.0 ## Steps To Reproduce 1. create-react-app test-render --template typescript 2. go to src/App.tsx write useEffect() in the function before return, such as "console.log('app') " in the callback 3. npm run start 4. checkout the console, you will find this function runs two times 5. if I use the React 17 ,there will be only once print <!-- Your bug will get fixed much faster if we can run your code and it doesn't have dependencies other than React. Issues without reproduction steps or code examples may be immediately closed as not actionable. --> Link to code example: https://github.com/Voiceu-zuixin/react18-test-render <!-- Please provide a CodeSandbox (https://codesandbox.io/s/new), a link to a			
24467	0	Other	24467		NONE		Support	Input Insufficient	Insufficient Sample Data
24938	0	Other	24938	Use of styled components ?	NONE		Support	Input Insufficient	
				React Lazy Load not working with variable path		I have the same issue when using Lazy Load with variable path. if I put the path in the variable, I see "Can't find the module" ... const filePath = `@pages/general/general`; const component = React.lazy(() => import(`\${filePath}`)); ... but if I put the path directly, it's working ... const component = React.lazy(() => import("@pages/general/general")); ... any suggestions? _Originally posted by https://github.com/facebook/react/issues/16132#issue-467840101			
25300	0	Other	25300		NONE		Support	Insufficient Sample Data	Input Insufficient

25520	0	Other	25520	<p>Bug: Github pages not displaying my Vite.js app</p>	<p>Link to code example: https://github.com/claudeMassaad/TENZIES-GAME-REACT-APP</p> <p>Here's my main and index.html in Vite app:</p> <p>![[Screen Shot 2022-10-20 at 10 16 31 AM]](https://user-images.githubusercontent.com/109232112/196881586-b1023f1c-5173-4200-9395-6c346833aec7.png)</p> <p>![[Screen Shot 2022-10-20 at 10 16 45 AM]](https://user-images.githubusercontent.com/109232112/196881627-064570e9-e384-40c1-a44a-6879da8eb0e2.png)</p> <p>## The current behavior</p> <p>![[Screen Shot 2022-10-20 at 10 15 39 AM]](https://user-images.githubusercontent.com/109232112/196881433-8e7d367b-4f93-406d-b262-88627627f5b7.png)</p> <p>Once i push to my github and then go to the pages feature on github where github generates a link for my web app, the screen is white and i get this error: Failed to load module script: Expected a JavaScript module script but the server responded with a MIME type of "text/jsx". Strict MIME type checking is enforced for module scripts per HTML spec.</p> <p>Now i cant change the name of main.jsx to main.js because Vite requires .jsx extensions.</p> <p>Any help?</p> <p>## The expected behavior</p> <p>I am running a react app created with Vite.js locally and everything works perfectly when i run npm run dev. It should do the same when github generates a link for my website</p>	Support	Insufficient Sample Data	Input Insufficient
-------	---	-------	-------	--	---	---------	--------------------------	--------------------

APPENDIX B – Jupyter Notebook Colab File PDF
(pySparkML & John Snow Labs Spark NLP)

Import the required packages and set up the Spark session with spark-nlp.

```
In [ ]: !wget http://setup.johnsnowlabs.com/colab.sh -O - | bash
```

```
In [ ]: import sparknlp
from sparknlp.base import *
from sparknlp.annotator import *
spark = sparknlp.start()

print("Spark NLP version: {}".format(sparknlp.version()))
print("Apache Spark version: {}".format(spark.version))
```

Spark NLP version: 4.2.5

Apache Spark version: 3.2.1

Load the CSV file (n=1000 samples) containing our manual labels as the target vector

```
In [ ]: from google.colab import files
uploaded = files.upload()
```

```
In [ ]: from pyspark.sql.functions import col
df = (spark.read
      .format("csv")
      .option("header", "true")
      .option("inferSchema", "true")
      .option("multiline", "true")
      .option("quote", "'")
      .option("escape", "\\")
      .option("escape", "'")
      .load("GH-React.csv")
      )
df = df.select(col('number'), col('title'), col('author_association'), col('bo
```

```
In [ ]: # a helper function to get the shape of a Spark DF
def sparkdf_shape(df):
    return df.count(), len(df.columns)
```

```
In [ ]: print("Shape:", sparkdf_shape(df))
```

Shape: (1000, 5)

Machine Learning Pipeline

Stage 1

Split the data into training (80%) and validation(20%) sets. We will stratify based on the label since our dataset is imbalanced.

```
In [ ]: # create a stratified sample for the training set using a 0.8 ratio
train = df.stat.sampleBy("Target", fractions={"Bug":0.8, "Feature":0.8, "Suppo
```

```
validate = df.exceptAll(train)
```

```
In [ ]: print("training set size:", train.count())
        print("validation set size:", validate.count())
```

```
training set size: 799
validation set size: 201
```

Stage 2

The `strip_text()` function is defined below. It takes in a String formatted as Markdown from GitHub and pre-processes it to return a new string ready for the next stages in our ML Pipeline.

```
In [ ]: import re
        from pyspark.sql.functions import udf

        @udf("String")
        def strip_text(text):
            if text is not None:
                stripped = text.lower()

                # remove all headings, bold text, and HTML comments from the Markdown text
                # These items have all been used by the React team in their issue templates
                headings_pattern = r'(<=\\s|^){1,6}(\\.\\?)*$'
                bold_pattern = r'\\*\\*(\\.\\?)*\\*\\*(?!\\*)'
                comments_pattern = r'<!--(\\.|\\n)*?-->'
                combined_pattern = r'|'.join((headings_pattern, bold_pattern, comments_pattern))

                stripped = re.sub(combined_pattern, '', stripped)

                # find all URLs in the string, and then remove the final directory from each
                # there may be useful patterns based on what URLs issues are commonly linked to
                url_pattern = re.compile(r'(https?:\\/\\[^\\s]+)')
                for url in re.findall(url_pattern, stripped):
                    new_url = url.rsplit("/", 1)[0]
                    stripped = stripped.replace(url, new_url)

                non_alpha_pattern = r'^A-Za-z ]+'
                stripped = re.sub(non_alpha_pattern, '', stripped)

                return ' '.join(stripped.split())
            else:
                return " "
```

Apply the `strip_text()` function to both the title and body columns in the train and validation datasets

```
In [ ]: train_data = train.withColumn("body", strip_text(col("body"))).withColumn("title", strip_text(col("title")))
        validation_data = validate.withColumn("body", strip_text(col("body"))).withColumn("title", strip_text(col("title")))
```

Check that the `strip_text()` function worked as expected on one sample:

```
In [ ]: train_data.take(1)
```

```
Out[ ]: [Row(number=11947, title='reactnativecustomtabs not return response', author_association='NONE', body='i have created button to open custontabs to use external url into reactnative app external url have some forms that are submitted and return array as a response into another page ie success page i want to get response from success page into app and customtabs would be close automatically code for custom taburl httpswwwexamplecomcustomtabsopenurlurl toolbarcolor d benableurlbarhiding trueshowpagetitle trueenableddefaultshare trueanimations animationsslidethenlaunched boolean consoleloglaunched custom tabs launchedcatcherr consoleerrorerr', Target='Feature')]
```

Stage 3

Create a TF-IDF features vector using a PySpark Pipeline. We will use TF-IDF applied to stemmed ngrams from both the issue title and body columns. We wi

We will apply this step separately to both the body and title to produce a different set of features for each. The tokens in the title may hold different importance than the same token in the body.

We will additionally add in the feature 'author_association' from the GitHub issue, as there may be a correlation between Members/Collaborators/Contributors submitting more valid bugs/feature requests than "None" users. This will be applied using one-hot-encoder.

```
In [ ]: from pyspark.ml.feature import CountVectorizer, IDF
from pyspark.ml.feature import StringIndexer
from pyspark.ml.feature import OneHotEncoder
from pyspark.ml.feature import VectorAssembler
from pyspark.ml import Pipeline
```

```
In [ ]: # NLP Pipeline
documentAssemblerTitle = DocumentAssembler().setInputCol('title').setOutputCol('document_title')
documentAssemblerBody = DocumentAssembler().setInputCol('body').setOutputCol('document_body')
tokenizer_titles = Tokenizer().setInputCols(['titles']).setOutputCol('tokenized_titles')
tokenizer_bodies = Tokenizer().setInputCols(['bodies']).setOutputCol('tokenized_bodies')
lemmatizer_titles = LemmatizerModel.pretrained().setInputCols(['tokenized_titles']).setOutputCol('lemmatized_titles')
lemmatizer_bodies = LemmatizerModel.pretrained().setInputCols(['tokenized_bodies']).setOutputCol('lemmatized_bodies')
ngrammer_titles = NGramGenerator().setInputCols(['trunc_titles']).setOutputCol('ngrams_titles')
ngrammer_bodies = NGramGenerator().setInputCols(['trunc_bodies']).setOutputCol('ngrams_bodies')
finisher = Finisher().setInputCols(['ngrams_titles', 'ngrams_bodies'])

tf_titles = CountVectorizer(inputCol='finished_ngrams_titles', outputCol='tf_features_titles')
tf_bodies = CountVectorizer(inputCol='finished_ngrams_bodies', outputCol='tf_features_bodies')
idf_titles = IDF(inputCol='tf_features_titles', outputCol='idf_titles', minDocF=1)
idf_bodies = IDF(inputCol='tf_features_bodies', outputCol='idf_bodies', minDocF=1)
author_stringIdx = StringIndexer(inputCol="author_association", outputCol="author_index")
ohe = OneHotEncoder(inputCol="author_index", outputCol="aa")
assembler = VectorAssembler(inputCols=['aa', 'idf_titles', 'idf_bodies'], outputCol='features')
label_stringIdx = StringIndexer(inputCol = "Target", outputCol = "label")

nlp_pipe = Pipeline().setStages([documentAssemblerTitle,
                                documentAssemblerBody,
                                tokenizer_titles,
                                tokenizer_bodies,
                                lemmatizer_titles,
```

```

lemmatizer_bodies,
ngrammer_titles,
ngrammer_bodies,
finisher,
tf_titles,
tf_bodies,
idf_titles,
idf_bodies,
author_stringIdx,
ohe,
assembler,
label_stringIdx])

```

lemma_antbnc download started this may take some time.
 Approximate size to download 907.6 KB
 [OK!]
 lemma_antbnc download started this may take some time.
 Approximate size to download 907.6 KB
 [OK!]

NOTE: Spark NLP and Spark ML are not compatible in terms of the pySpark Pipeline and CrossValidate (took a very long time to learn this). We need to make the NLP pre-processing and intermediate step in the process.

Source: <https://github.com/JohnSnowLabs/spark-nlp/issues/1158>

We will fit/transform the NLP part of the pipeline and feed it into the Classifiers later on. This also has the added benefit of not having to re-run the NLP pre-processing on each iteration of grid search, but load it from cache instead.

```

In [ ]: nlp_fit = nlp_pipe.fit(train_data)
        nlp_train = nlp_fit.transform(train_data)
        # capturing full dataframe here to use for figs later
        nlp_train_full = nlp_train
        nlp_train = nlp_train.select(col('features'), col('label'))

        nlp_validate = nlp_fit.transform(validation_data)
        nlp_validate = nlp_validate.select(col('number'), col('features'), col('label'))

        # Save to file for future reference, and load to cache
        nlp_train.write.mode('overwrite').parquet("./nlp/nlp_train.parquet")
        nlp_train = spark.read.parquet("./nlp/nlp_train.parquet")
        nlp_train.cache()

        nlp_validate.write.mode('overwrite').parquet("./nlp/nlp_validate.parquet")
        nlp_validate = spark.read.parquet("./nlp/nlp_validate.parquet")
        nlp_validate.cache()

```

Out[]: DataFrame[number: int, features: vector, label: double]

Plotting / helper functions

```

In [ ]: # bar/count plot helper function

```

```

import seaborn as sns
import matplotlib.pyplot as plt

def term_freq_plot(df, title):
    dfp = df.toPandas()
    sns.set_theme(style="ticks")
    plt.figure(figsize=(15,8))
    freq_plot = sns.barplot(data=dfp, x=dfp.columns[0], y=dfp.columns[1], hue=Noi
    freq_plot.set(title=title)
    for label in freq_plot.get_xticklabels():
        label.set_rotation(45)

```

```

In [ ]: # confusion matrix plotting

import matplotlib.pyplot as plt
import seaborn as sns
from pyspark.mllib.evaluation import MulticlassMetrics
import warnings

def confusion_matrix_plotter(df_model_output, model_name, target_label='label'

    warnings.filterwarnings("ignore")

    if(nparray == 'n'):
        predications_and_labels = df_model_output.select(col(predict_label),col(ta
        # predications_and_labels.show(5)
        metrics = MulticlassMetrics(predications_and_labels.rdd)
        confusion_matrix = metrics.confusionMatrix().toArray()
    else:
        confusion_matrix=df_model_output
        print(confusion_matrix)
        plt.figure(figsize=(10,6))
        fx=sns.heatmap(confusion_matrix, annot=True, fmt=".1f",cmap="GnBu", xticklabel
        fx.set_title('Confusion Matrix Results %(model_name)s \n' %{"model_name": mo
        fx.set_xlabel('\n Predicted Values\n')
        fx.set_ylabel('Actual Values\n');
        fx.xaxis.set_ticklabels(["Other", "Support", "Bug", "Feature"])
        fx.yaxis.set_ticklabels(["Other", "Support", "Bug", "Feature"])
        if(reverse_xaxis == 'y'):
            fx.invert_xaxis()
        plt.show()

```

```

In [ ]: # logistic regression heatmap plotter

import re
import pandas as pd

def logReg_cv_scores_heatmap(param_grid, model):
    plt.figure(figsize=(10,6))

    i = 0
    reg_Param = []
    elastic_NetParam = []

    while i < (len(param_grid)):
        result = re.search(r"(regParam).*(?: (\d+.\d+),).+(elasticNetParam).+(:.*('
        reg_Param.append(result.groups()[2])

```

```

        elastic_NetParam.append(result.groups()[5])
        i += 1

    # print(reg_Param)
    # print(elastic_NetParam)
    # print(model.avgMetrics)

    data_lists = list(zip(reg_Param, elastic_NetParam, model.avgMetrics))

    df_labels = ["regParam", "elasticNetParam", "Score"]

    data_df = spark.createDataFrame(data=data_lists, schema=df_labels)
    # data_df.show()

    heatmap_df = data_df.toPandas().pivot("regParam", "elasticNetParam", "Score")
    # heatmap_df.head(10)

    plot = sns.heatmap(heatmap_df, annot=True, fmt=".2f", linewidths=0.25, cmap =
    plot.invert_yaxis()

    return heatmap_df

```

```

In [ ]: # Naive Bayes heatmap plotter (similar to logistic regression func above
# - at this time, I just adapted the previous instead of making a more general

import re
import pandas as pd

def nb_cv_scores_heatmap(param_grid, model):
    plt.figure(figsize=(10,6))

    i = 0
    param1 = []
    param2 = []

    while i < (len(param_grid)):
        result = re.search(r"name=.(smoothing).*:(\d+.\d+),.(modelType).+"):
        # print(result.groups())
        param1.append(result.groups()[1])
        param2.append(result.groups()[3])
        i += 1

    # print(param1)
    # print(param1)
    # print(model.avgMetrics)

    data_lists = list(zip(param1, param2, model.avgMetrics))

    df_labels = ["smoothing", "modelType", "Score"]

    data_df = spark.createDataFrame(data=data_lists, schema=df_labels)
    data_df.show()

    heatmap_df = data_df.toPandas().pivot("smoothing", "modelType", "Score")
    heatmap_df.head(10)

    plot = sns.heatmap(heatmap_df, annot=True, linewidths=0.25, cmap = sns.cm.ro

```

```

plot.invert_yaxis()

return heatmap_df

```

Title & Body (finished ngram) Figures

```

In [ ]: from pyspark.sql.types import StringType

df_finished_ngrams_titles = nlp_train_full.select('finished_ngrams_titles').rdd
# print(df_finished_ngrams_titles.count())
# df_finished_ngrams_titles.show(5)

df_finished_ngrams_bodies = nlp_train_full.select('finished_ngrams_bodies').rdd
# print(df_finished_ngrams_bodies.count())
# df_finished_ngrams_bodies.show(5)

term_freq_titles = df_finished_ngrams_titles.rdd.countByValue()
term_freq_titles_df = pd.DataFrame({'term': list(term_freq_titles.keys()), 'frequency': list(term_freq_titles.values())})
title_freqs = spark.createDataFrame(term_freq_titles_df).orderBy('frequency', ascending=False)
title_freqs.show(3)
print(sparkdf_shape(title_freqs))

term_freq_body = df_finished_ngrams_bodies.rdd.countByValue()
term_freq_body_df = pd.DataFrame({'term': list(term_freq_body.keys()), 'frequency': list(term_freq_body.values())})
body_freqs = spark.createDataFrame(term_freq_body_df).orderBy('frequency', ascending=False)
body_freqs.show(3)
print(sparkdf_shape(body_freqs))

+-----+-----+
| term|frequency|
+-----+-----+
| {bug}|      221|
| {in}|      197|
| {be}|      166|
+-----+-----+
only showing top 3 rows

(5899, 2)

+-----+-----+
| term|frequency|
+-----+-----+
| {the}|      4169|
| {be}|      2875|
| {to}|      2435|
+-----+-----+
only showing top 3 rows

(59550, 2)

```

```

In [ ]: import pyspark.sql.functions as F
df_title_freqs = title_freqs.select(F.col("term.terms").alias("term"), F.col("frequency").alias("frequency"))
df_title_freqs.show()

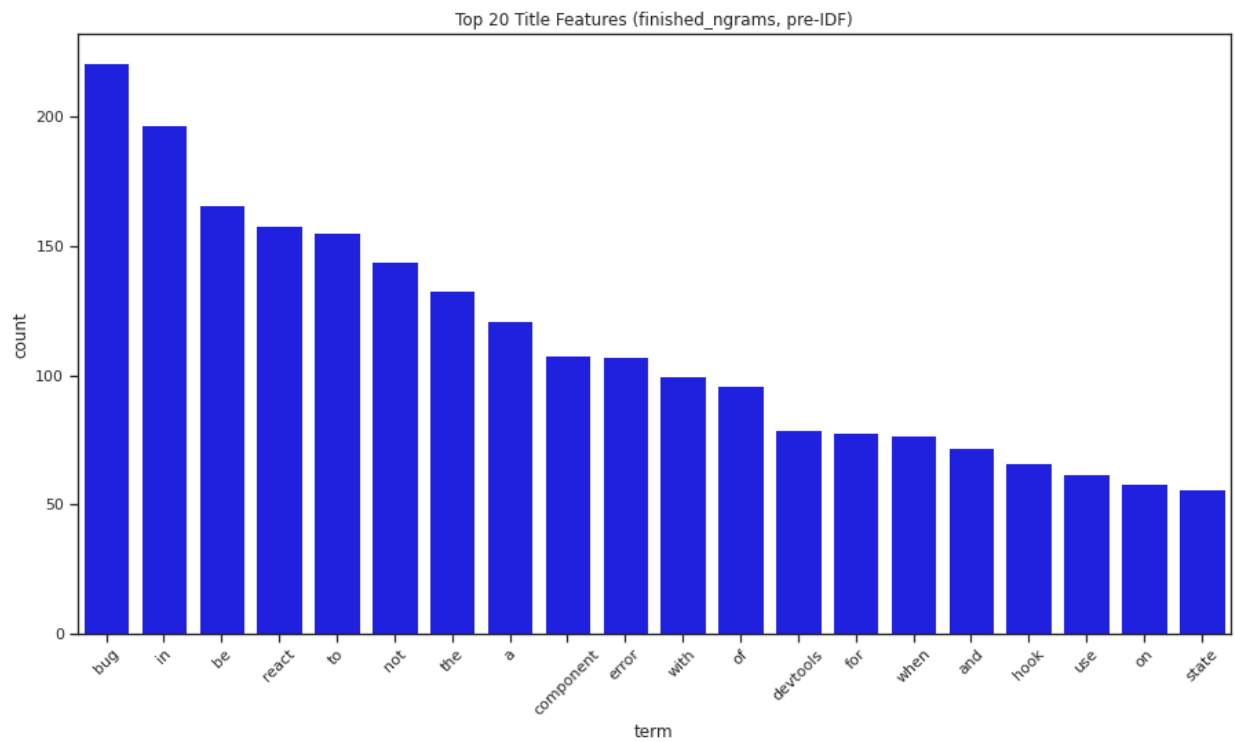
df_body_freqs = body_freqs.select(F.col("term.terms").alias("term"), F.col("frequency").alias("frequency"))
df_body_freqs.show()

```

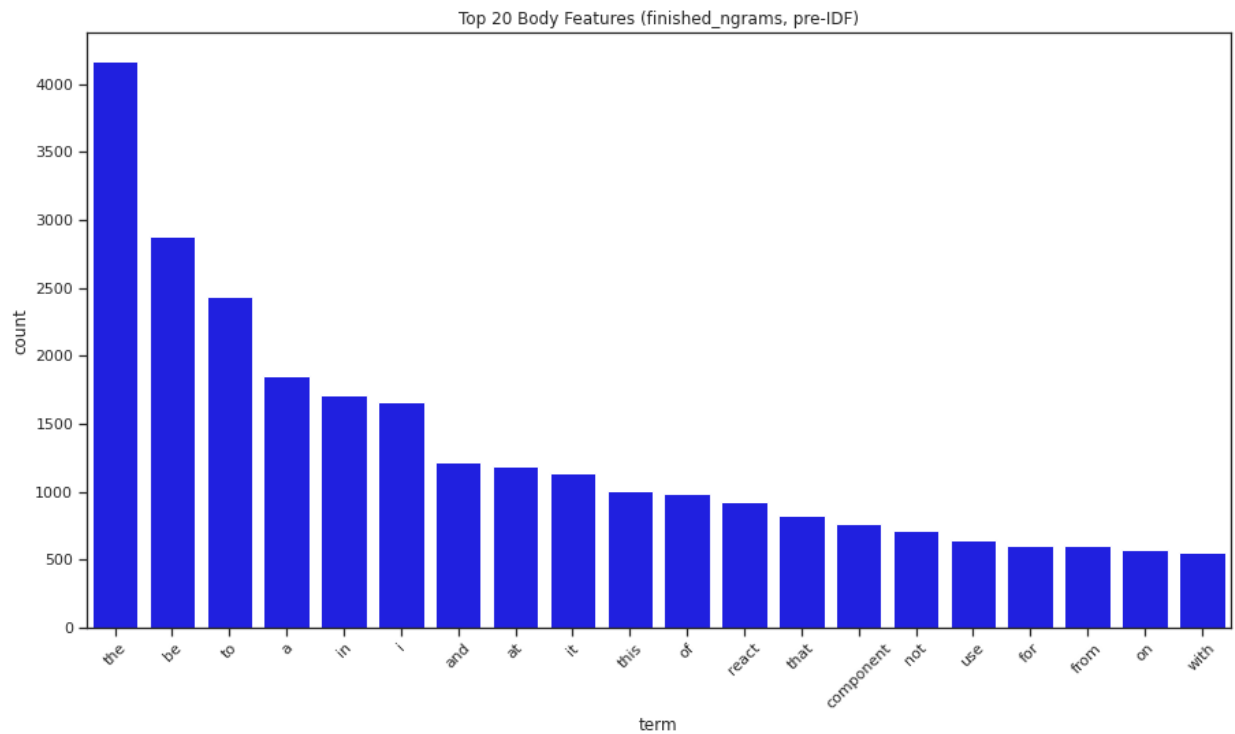

term	count
bug	221
in	197
be	166
react	158
to	155
not	144
the	133
a	121
component	108
error	107
with	100
of	96
devtools	79
for	78
when	77
and	72
hook	66
use	62
on	58
state	56

term	count
the	4169
be	2875
to	2435
a	1851
in	1707
i	1663
and	1221
at	1190
it	1138
this	1008
of	988
react	925
that	821
component	768
not	713
use	642
for	600
from	599
on	576
with	558

```
In [ ]: term_freq_plot(df_title_freqs, "Top 20 Title Features (finished_ngrams, pre-ID)
```



```
In [ ]: term_freq_plot(df_body_freqs, "Top 20 Body Features (finished_ngrams, pre-IDF)")
```



Stage 4

Apply and fit the model pipeline.

First, fit and transform the pipeline with our training dataset using Logistic Regression classifier.

```
In [ ]: from pyspark.ml.classification import LogisticRegression
```

```
lr = LogisticRegression()
lr_model = lr.fit(nlp_train)
```

```
In [ ]: predictions_train = lr_model.transform(nlp_train)
```

Evaluate the predictions. We have a perfect training score, which could be expected given that we have many more features than samples on a linear model.

```
In [ ]: from pyspark.ml.evaluation import MulticlassClassificationEvaluator

evaluator = MulticlassClassificationEvaluator()

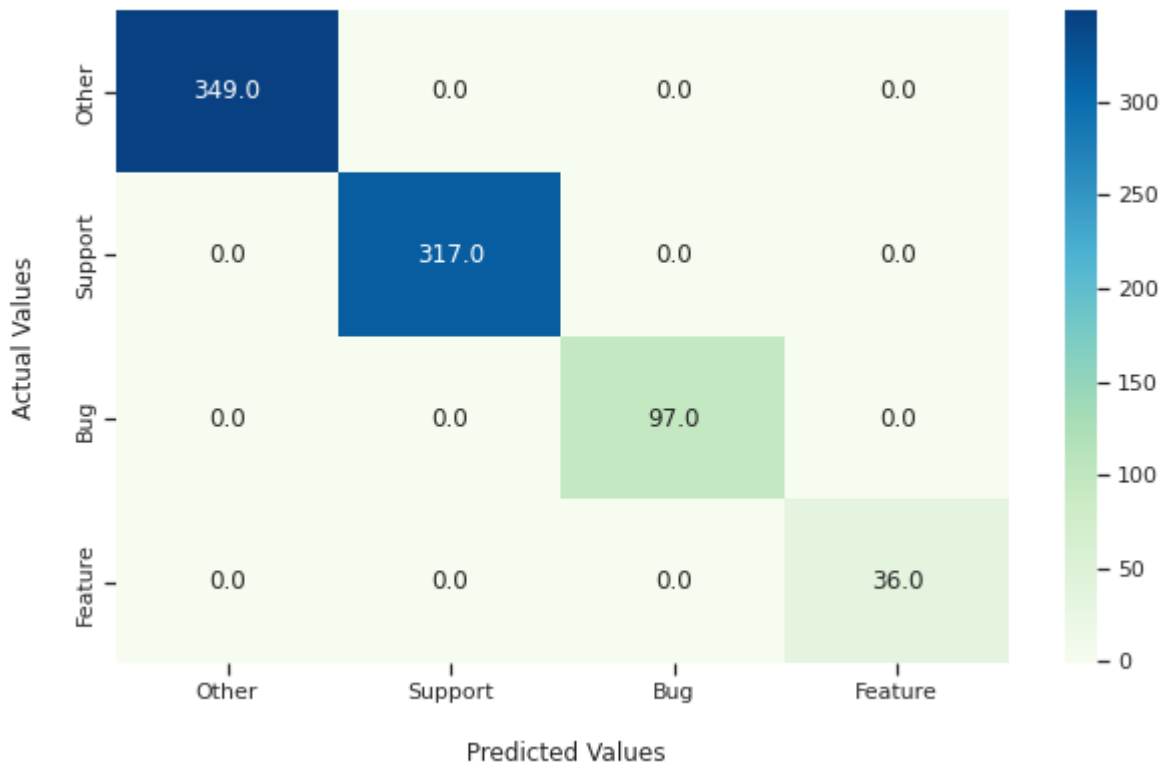
accuracy = predictions_train.filter(predictions_train.label == predictions_train.prediction)
score = evaluator.evaluate(predictions_train)
print("Train Accuracy Score: {0:.4f}".format(accuracy))
print("Train {0}: {1:.4f}".format(evaluator.getMetricName(), score))
```

```
Train Accuracy Score: 1.0000
Train f1: 1.0000
```

```
In [ ]: # call helper to plot confusion matrix
confusion_matrix_plotter(predictions_train, "LR Model")
```

```
[[349.  0.  0.  0.]
 [ 0. 317.  0.  0.]
 [ 0.  0. 97.  0.]
 [ 0.  0.  0. 36.]]
```

Confusion Matrix Results LR Model



Save the trained model for faster loading in the future:

```
In [ ]: from pyspark.ml.classification import LogisticRegressionModel
lr_model.write().overwrite().save("./trainedmodels/lr")
lr_model = LogisticRegressionModel.load("./trainedmodels/lr")
```

Now, apply the model to our validation set to see how it performs on new data:

```
In [ ]: predictions_validate = lr_model.transform(nlp_validate)
```

```
In [ ]: accuracy = predictions_validate.filter(predictions_validate.label == prediction)
score = evaluator.evaluate(predictions_validate)
print("Validation Accuracy Score: {0:.4f}".format(accuracy))
print("Validation {0}: {1:.4f}".format(evaluator.getMetricName(), score))
```

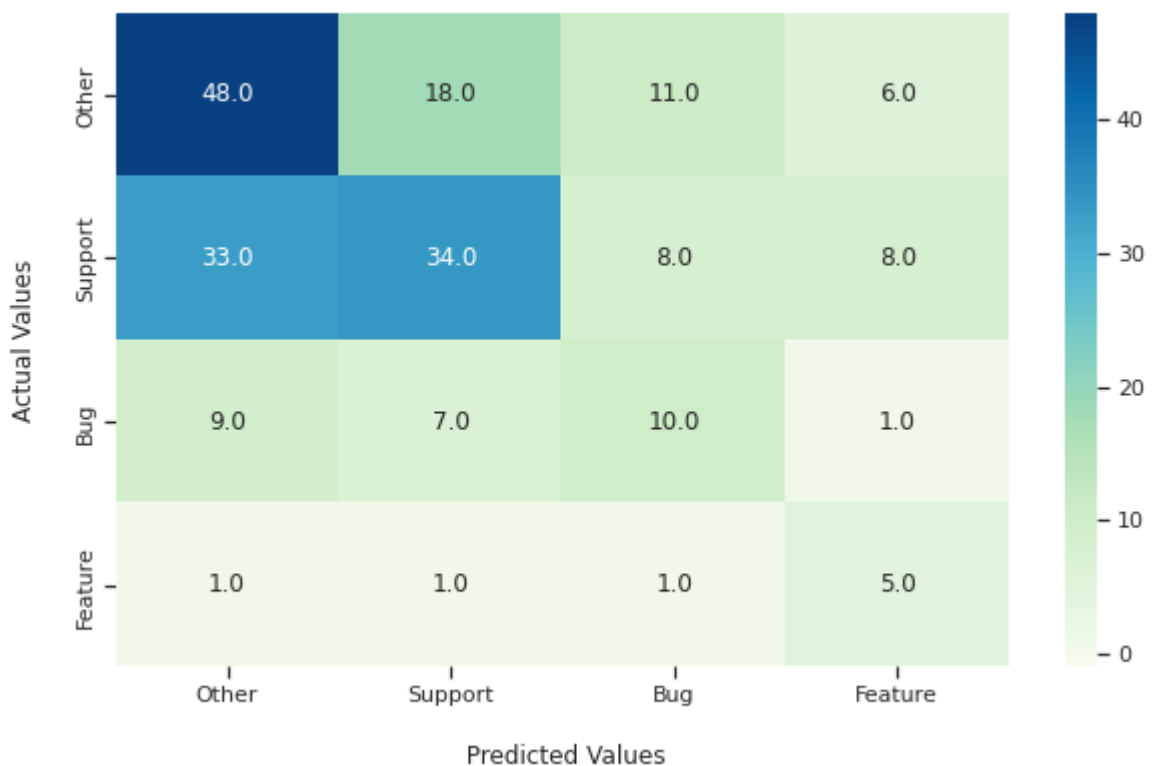
Validation Accuracy Score: 0.4826

Validation f1: 0.4855

```
In [ ]: # call helper to plot confusion matrix
confusion_matrix_plotter(predictions_validate, "LR Model")
```

```
[[48. 18. 11.  6.]
 [33. 34.  8.  8.]
 [ 9.  7. 10.  1.]
 [ 1.  1.  1.  5.]]
```

Confusion Matrix Results LR Model



Take a look at the confusion matrix to see where the model made its mistakes:

```
In [ ]: # generate the hits/misses
confusion_table = predictions_validate.groupBy('label', 'prediction').count()

# Get the labels back from the model
from pyspark.ml.feature import IndexToString
```

```

t_labels = IndexToString(inputCol="label", outputCol="TargetLabel")
p_labels = IndexToString(inputCol='prediction', outputCol="PredictionLabel", l

confusion_table = t_labels.transform(confusion_table)
confusion_table = p_labels.transform(confusion_table)
confusion_table = confusion_table.select(col("TargetLabel"), col("PredictionLabel"))

#display the table with the confusion results
confusion_table.show()

```

```

+-----+-----+-----+
|TargetLabel|PredictionLabel|count|
+-----+-----+-----+
|      Bug|      Other|    9|
|   Support|   Support|   34|
|   Feature|      Bug|    1|
|      Other|   Support|   18|
|      Bug|      Bug|   10|
|   Support|      Other|   33|
|   Feature|   Support|    1|
|      Bug|   Feature|    1|
|      Bug|   Support|    7|
|   Support|      Bug|    8|
|      Other|      Other|   48|
|   Support|   Feature|    8|
|      Other|      Bug|   11|
|   Feature|   Feature|    5|
|      Other|   Feature|    6|
|   Feature|      Other|    1|
+-----+-----+-----+

```

Stage 5

Grid Search for optimizing model

Logistic Regression

Use grid search with cross validate to find potentially better model parameters.

We will use the lr (LogisticRegression classifier) we created earlier

```

In [ ]: from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
import numpy as np

paramGrid_lr = ParamGridBuilder() \
    .addGrid(lr.regParam, [0, 0.01, 0.1, 1, 10]) \
    .addGrid(lr.elasticNetParam, [0, 1]) \
    .build()

# MCE metricName="f1" (default scoring metric)
evaluator = MulticlassClassificationEvaluator()

```

```
cv = CrossValidator(estimator=lr,
                    estimatorParamMaps=paramGrid_lr,
                    evaluator=evaluator,
                    numFolds=5,
                    seed=9)
```

```
# use the output from the NLP train pipeline
cvModel = cv.fit(nlp_train)
```

```
In [ ]: # save the model to make future analysis easier (won't have to re-perform grid
from pyspark.ml.tuning import CrossValidatorModel
cvModel.write().overwrite().save("./trainedmodels/cvlr")
cvModel = CrossValidatorModel.load("./trainedmodels/cvlr")
```

```
In [ ]: # the parameters for the best model:
cvModel.bestModel.extractParamMap()
```

```
Out[ ]: {Param(parent='LogisticRegression_7c337dfc6106', name='aggregationDepth', doc
='suggested depth for treeAggregate (>= 2).'): 2,
  Param(parent='LogisticRegression_7c337dfc6106', name='elasticNetParam', doc
='the ElasticNet mixing parameter, in range [0, 1]. For alpha = 0, the penalty
is an L2 penalty. For alpha = 1, it is an L1 penalty.'): 0.0,
  Param(parent='LogisticRegression_7c337dfc6106', name='family', doc='The name
of family which is a description of the label distribution to be used in the m
odel. Supported options: auto, binomial, multinomial'): 'auto',
  Param(parent='LogisticRegression_7c337dfc6106', name='featuresCol', doc='feat
ures column name.'): 'features',
  Param(parent='LogisticRegression_7c337dfc6106', name='fitIntercept', doc='whe
ther to fit an intercept term.'): True,
  Param(parent='LogisticRegression_7c337dfc6106', name='labelCol', doc='label c
olumn name.'): 'label',
  Param(parent='LogisticRegression_7c337dfc6106', name='maxBlockSizeInMB', doc
='maximum memory in MB for stacking input data into blocks. Data is stacked wi
thin partitions. If more than remaining data size in a partition then it is ad
justed to the data size. Default 0.0 represents choosing optimal value, depend
s on specific algorithm. Must be >= 0.'): 0.0,
  Param(parent='LogisticRegression_7c337dfc6106', name='maxIter', doc='max numb
er of iterations (>= 0.'): 100,
  Param(parent='LogisticRegression_7c337dfc6106', name='predictionCol', doc='pr
ediction column name.'): 'prediction',
  Param(parent='LogisticRegression_7c337dfc6106', name='probabilityCol', doc='C
olumn name for predicted class conditional probabilities. Note: Not all models
output well-calibrated probability estimates! These probabilities should be tr
eated as confidences, not precise probabilities.'): 'probability',
  Param(parent='LogisticRegression_7c337dfc6106', name='rawPredictionCol', doc
='raw prediction (a.k.a. confidence) column name.'): 'rawPrediction',
  Param(parent='LogisticRegression_7c337dfc6106', name='regParam', doc='regular
ization parameter (>= 0.'): 0.1,
  Param(parent='LogisticRegression_7c337dfc6106', name='standardization', doc
='whether to standardize the training features before fitting the model.'): Tr
ue,
  Param(parent='LogisticRegression_7c337dfc6106', name='threshold', doc='Thresh
old in binary classification prediction, in range [0, 1]. If threshold and thr
esholds are both set, they must match.e.g. if threshold is p, then thresholds
must be equal to [1-p, p].'): 0.5,
  Param(parent='LogisticRegression_7c337dfc6106', name='tol', doc='the converge
nce tolerance for iterative algorithms (>= 0.'): 1e-06}
```

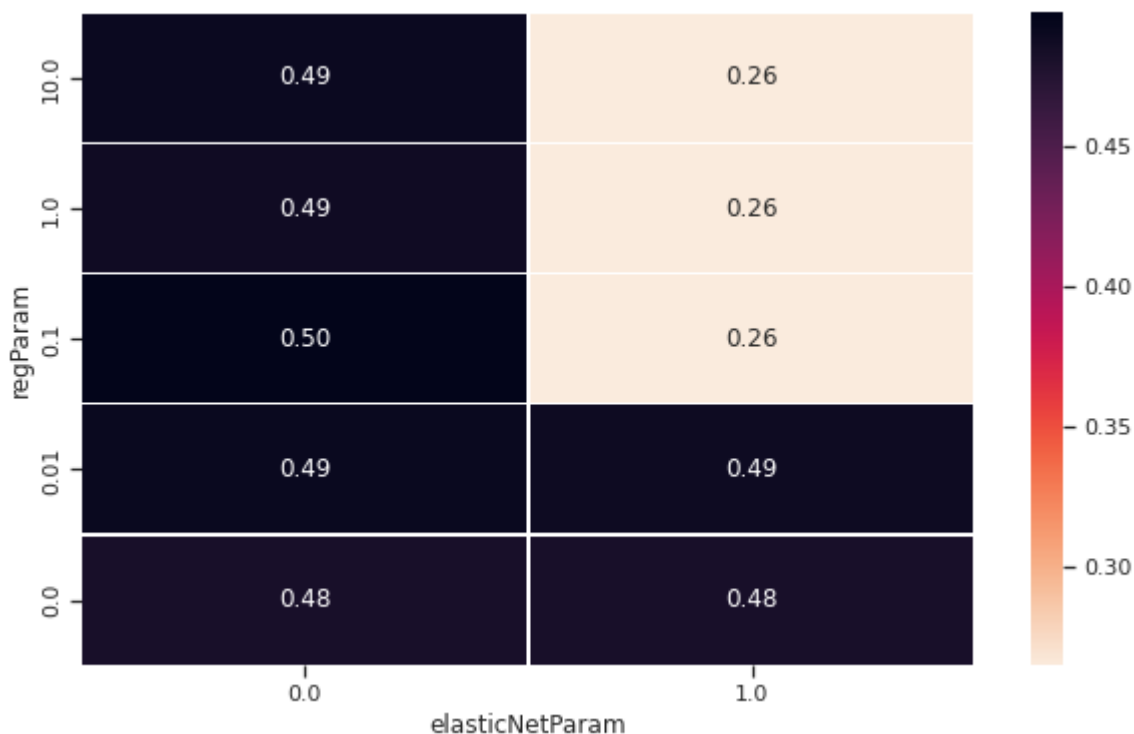
```
In [ ]: # get the accuracy metrics for the models. This is a list.
avgMetricsGrid = cvModel.avgMetrics
print(avgMetricsGrid)
np.mean(cvModel.avgMetrics)
```

[0.4829413780903705, 0.4829413780903705, 0.49170877529621726, 0.4891459129227088, 0.4978882781559933, 0.2648432438321038, 0.48874479547318617, 0.2648432438321038, 0.4907794028896253, 0.2648432438321038]

Out[]: 0.42186796524147835

```
In [ ]: # call helper to plot heatmap
heatmap_df = logReg_cv_scores_heatmap(paramGrid_lr, cvModel)
heatmap_df
```

```
Out[ ]: elasticNetParam      0.0      1.0
      regParam
      0.0  0.482941  0.482941
      0.01 0.491709  0.489146
      0.1  0.497888  0.264843
      1.0  0.488745  0.264843
     10.0  0.490779  0.264843
```



Now, we will fit our best model on the validation data again:

```
In [ ]: predictions_best_model_lr = cvModel.transform(nlp_validate)
```

```
In [ ]: accuracy = predictions_best_model_lr.filter(predictions_best_model_lr.label ==
score = evaluator.evaluate(predictions_best_model_lr)
```

```
print("Best Validation Accuracy Score (LR): {0:.4f}".format(accuracy))
print("Best Validation {0} (LR): {1:.4f}".format(evaluator.getMetricName(), score))
```

Best Validation Accuracy Score (LR): 0.5274
Best Validation f1 (LR): 0.4996

```
In [ ]: # generate the hits/misses
confusion_table = predictions_best_model_lr.groupBy('label','prediction').count()

# Get the labels back from the model
from pyspark.ml.feature import IndexToString
t_labels = IndexToString(inputCol="label", outputCol="TargetLabel")
p_labels = IndexToString(inputCol='prediction', outputCol="PredictionLabel", labelCol="label")

confusion_table = t_labels.transform(confusion_table)
confusion_table = p_labels.transform(confusion_table)
confusion_table = confusion_table.select(col("TargetLabel"), col("PredictionLabel"), col("count"))

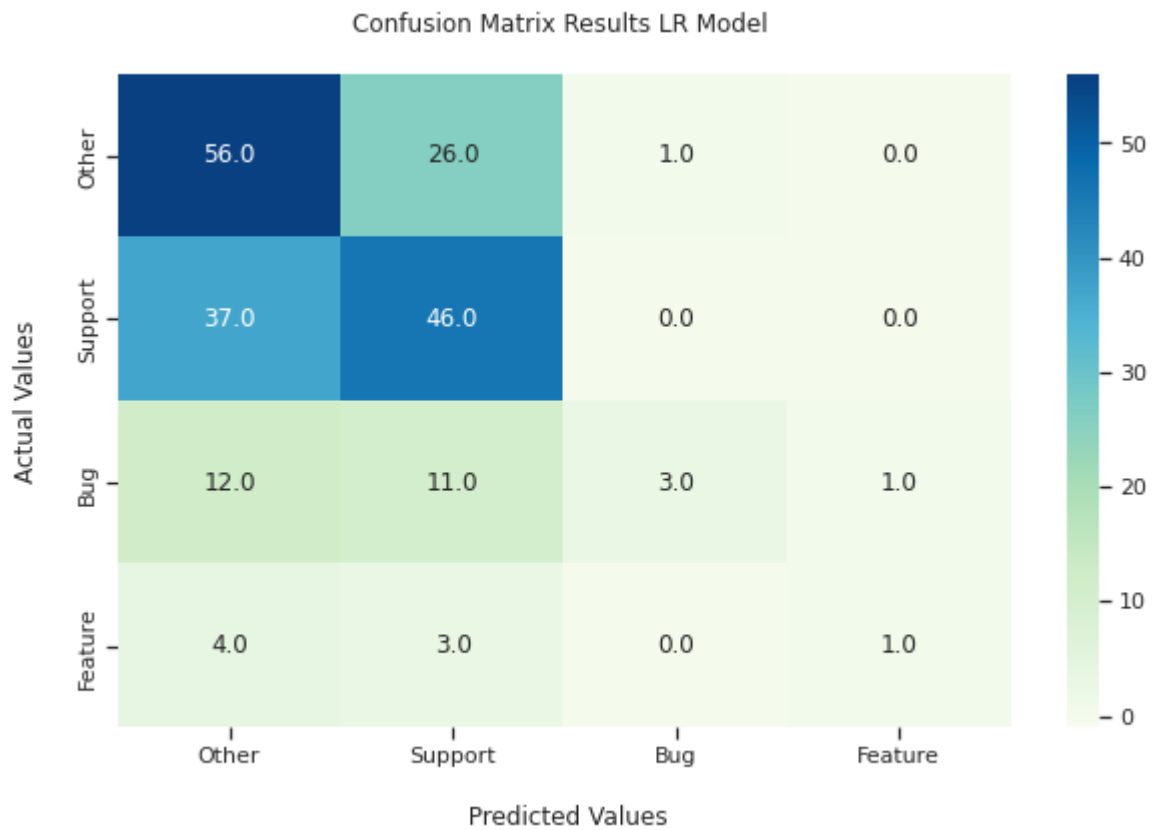
#display the table with the confusion results
confusion_table.show()
```

TargetLabel	PredictionLabel	count
Bug	Other	12
Support	Support	46
Other	Support	26
Support	Other	37
Bug	Bug	3
Feature	Support	3
Bug	Feature	1
Bug	Support	11
Other	Other	56
Other	Bug	1
Feature	Feature	1
Feature	Other	4

A few more metrics:

```
In [ ]: # call helper function to plot confusion matrix
confusion_matrix_plotter(predictions_best_model_lr, "LR Model")

[[56. 26.  1.  0.]
 [37. 46.  0.  0.]
 [12. 11.  3.  1.]
 [ 4.  3.  0.  1.]]
```

```
In [ ]: labels = ["Other", "Support", "Bug", "Feature"]
metrics = MulticlassMetrics(predictions_best_model_lr.select(col('label'),col(
print("Accuracy: ", metrics.accuracy)
for idx, label in enumerate(labels):
    print("Recall for {}: {}".format(label, metrics.precision(idx)))
    print("Precision for {}: {}".format(label, metrics.recall(idx)))
    print("f1 score for {}: {}".format(label, metrics.fMeasure(float(idx), 1.0))

Accuracy: 0.527363184079602
Recall for Other: 0.6746987951807228
Precision for Other: 0.5137614678899083
f1 score for Other: 0.5833333333333333
Recall for Support: 0.5542168674698795
Precision for Support: 0.5348837209302325
f1 score for Support: 0.5443786982248521
Recall for Bug: 0.11111111111111111
Precision for Bug: 0.75
f1 score for Bug: 0.19354838709677416
Recall for Feature: 0.125
Precision for Feature: 0.5
f1 score for Feature: 0.2
```

```
In [ ]: predictions_best_model_lr.select(col('label'),col('prediction')).groupBy("labe"
```

label	prediction	sum(label)	sum(prediction)
2.0	0.0	24.0	0.0
1.0	1.0	46.0	46.0
0.0	1.0	0.0	26.0
1.0	0.0	37.0	0.0
2.0	2.0	6.0	6.0
3.0	1.0	9.0	3.0
2.0	3.0	2.0	3.0
2.0	1.0	22.0	11.0
0.0	0.0	0.0	0.0
0.0	2.0	0.0	2.0
3.0	3.0	3.0	3.0
3.0	0.0	12.0	0.0

Naive Bayes

Create a new pipeline for Naive Bayes

```
In [ ]: from pyspark.ml.classification import NaiveBayes
        from pyspark.ml.evaluation import MulticlassClassificationEvaluator
        from pyspark.ml.tuning import CrossValidator, ParamGridBuilder

        nb = NaiveBayes()

        paramGrid_nb = ParamGridBuilder() \
            .addGrid(nb.smoothing, [0.0, 0.2, 0.4, 0.6, 0.8, 1.0]) \
            .addGrid(nb.modelType, ["multinomial", "complement", "gaussian"]) \
            .build()

        # MCE metricName="f1" (default scoring metric)
        evaluator = MulticlassClassificationEvaluator()

        cv_nb = CrossValidator(estimator=nb,
                               estimatorParamMaps=paramGrid_nb,
                               evaluator=evaluator,
                               numFolds=5,
                               seed=9)

        cvModelNB = cv_nb.fit(nlp_train)
```

```
In [ ]: # save the model to make future analysis easier (won't have to re-perform grid
        from pyspark.ml.tuning import CrossValidatorModel
        cvModelNB.write().overwrite().save("./trainedmodels/cvnb")
        cvModelNB = CrossValidatorModel.load("./trainedmodels/cvnb")
```

```
In [ ]: # the parameters for the best model:
        cvModelNB.bestModel.extractParamMap()
```

```
Out[ ]: {Param(parent='NaiveBayes_2e69d3d8e46e', name='featuresCol', doc='features column name.'): 'features',
  Param(parent='NaiveBayes_2e69d3d8e46e', name='labelCol', doc='label column name.'): 'label',
  Param(parent='NaiveBayes_2e69d3d8e46e', name='modelType', doc='The model type which is a string (case-sensitive). Supported options: multinomial (default), bernoulli and gaussian.'): 'complement',
  Param(parent='NaiveBayes_2e69d3d8e46e', name='predictionCol', doc='prediction column name.'): 'prediction',
  Param(parent='NaiveBayes_2e69d3d8e46e', name='probabilityCol', doc='Column name for predicted class conditional probabilities. Note: Not all models output well-calibrated probability estimates! These probabilities should be treated as confidences, not precise probabilities.'): 'probability',
  Param(parent='NaiveBayes_2e69d3d8e46e', name='rawPredictionCol', doc='raw prediction (a.k.a. confidence) column name.'): 'rawPrediction',
  Param(parent='NaiveBayes_2e69d3d8e46e', name='smoothing', doc='The smoothing parameter, should be >= 0, default is 1.0'): 0.2}
```

```
In [ ]: # get the accuracy metrics for the models. This is a list.
```

```
import numpy as np
avgMetricsGridNB = cvModelNB.avgMetrics
print(avgMetricsGridNB)
np.mean(cvModelNB.avgMetrics)
```

```
[0.39886667303320333, 0.390487351286241, 0.4488035265700841, 0.4793758467551451, 0.5049638854590692, 0.4488035265700841, 0.48153822085517206, 0.49683112073323443, 0.4488035265700841, 0.4797111380509994, 0.4927766919293613, 0.4488035265700841, 0.48111397515063536, 0.49041983094462505, 0.4488035265700841, 0.4782675250964634, 0.48437077854641253, 0.4488035265700841]
```

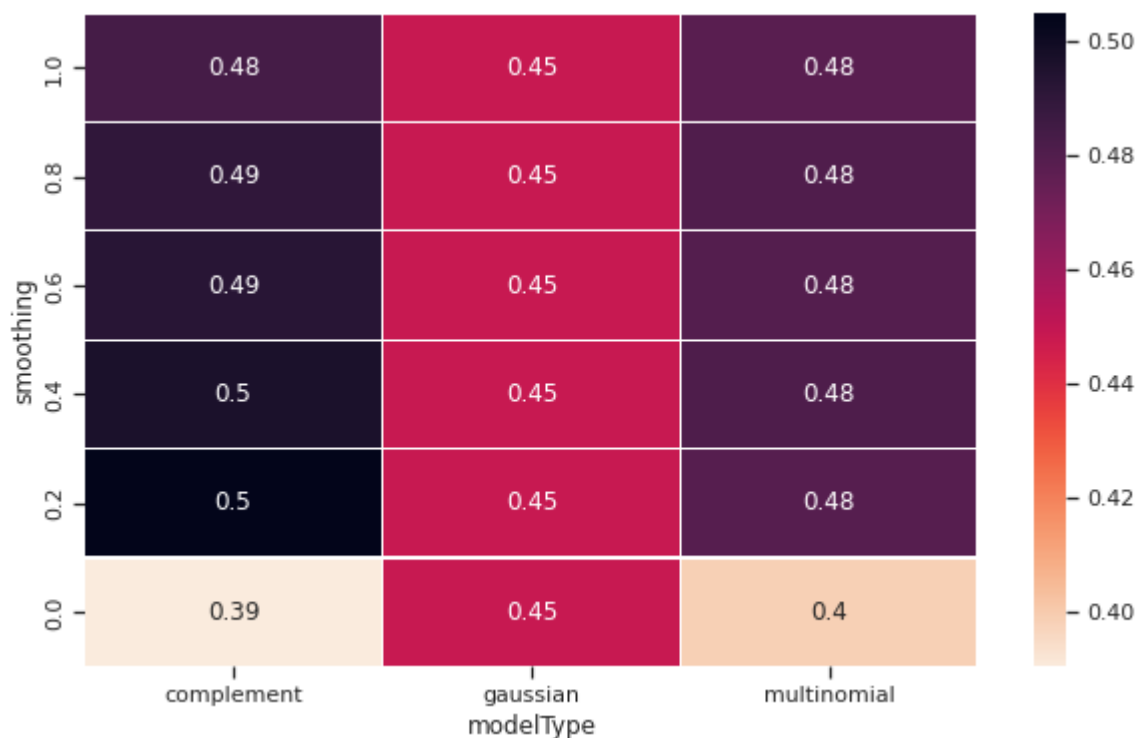
```
Out[ ]: 0.46397467762561484
```

```
In [ ]: heatmap_nb_df = nb_cv_scores_heatmap(paramGrid_nb, cvModelNB)
heatmap_nb_df
```

```
+-----+-----+-----+
|smoothing| modelType|          Score|
+-----+-----+-----+
|      0.0|multinomial|0.39886667303320333|
|      0.0| complement| 0.390487351286241|
|      0.0|  gaussian| 0.4488035265700841|
|      0.2|multinomial| 0.4793758467551451|
|      0.2| complement| 0.5049638854590692|
|      0.2|  gaussian| 0.4488035265700841|
|      0.4|multinomial|0.48153822085517206|
|      0.4| complement|0.49683112073323443|
|      0.4|  gaussian| 0.4488035265700841|
|      0.6|multinomial| 0.4797111380509994|
|      0.6| complement| 0.4927766919293613|
|      0.6|  gaussian| 0.4488035265700841|
|      0.8|multinomial|0.48111397515063536|
|      0.8| complement|0.49041983094462505|
|      0.8|  gaussian| 0.4488035265700841|
|      1.0|multinomial| 0.4782675250964634|
|      1.0| complement|0.48437077854641253|
|      1.0|  gaussian| 0.4488035265700841|
+-----+-----+-----+
```

```
Out[ ]: modelType complement gaussian multinomial
smoothing
```

0.0	0.390487	0.448804	0.398867
0.2	0.504964	0.448804	0.479376
0.4	0.496831	0.448804	0.481538
0.6	0.492777	0.448804	0.479711
0.8	0.490420	0.448804	0.481114
1.0	0.484371	0.448804	0.478268



```
In [ ]: predictions_best_model_nb = cvModelNB.transform(nlp_validate)
```

```
In [ ]: accuracy = predictions_best_model_nb.filter(predictions_best_model_nb.label ==
score = evaluator.evaluate(predictions_best_model_nb)
print("Best Validation Accuracy Score (NB): {0:.4f}".format(accuracy))
print("Best Validation {0} (NB): {1:.4f}".format(evaluator.getMetricName(), score))
```

```
Best Validation Accuracy Score (NB): 0.5075
Best Validation f1 (NB): 0.4699
```

```
In [ ]: # generate the hits/misses
confusion_table_nb = predictions_best_model_nb.groupBy('label', 'prediction').count()

# Get the labels back from the model
from pyspark.ml.feature import IndexToString
t_labels = IndexToString(inputCol="label", outputCol="TargetLabel")
p_labels = IndexToString(inputCol='prediction', outputCol="PredictionLabel", labelCol="label")

confusion_table_nb = t_labels.transform(confusion_table_nb)
confusion_table_nb = p_labels.transform(confusion_table_nb)
confusion_table_nb = confusion_table_nb.select(col("TargetLabel"), col("PredictionLabel"), col("count"))
```

```
#display the table with the confusion results
confusion_table_nb.show()
```

```
+-----+-----+-----+
|TargetLabel|PredictionLabel|count|
+-----+-----+-----+
|      Bug|      Other|    6|
|    Support|    Support|   67|
|      Other|    Support|   46|
|    Support|      Other|   12|
|      Bug|      Bug|    2|
|    Feature|    Support|    7|
|      Bug|    Support|   19|
|    Support|      Bug|    3|
|      Other|      Other|   32|
|    Support|    Feature|    1|
|      Other|      Bug|    3|
|    Feature|    Feature|    1|
|      Other|    Feature|    2|
+-----+-----+-----+
```

Some More Metrics:

```
In [ ]: # generate confusion matrix w/ sklearn due to pyspark glitch (dropping 'Feature'
from sklearn.metrics import confusion_matrix
confusion_table_nb_SKL = predictions_best_model_nb

from pyspark.ml.feature import IndexToString
t_labels = IndexToString(inputCol="label", outputCol="TargetLabel")
p_labels = IndexToString(inputCol='prediction', outputCol="PredictionLabel", l

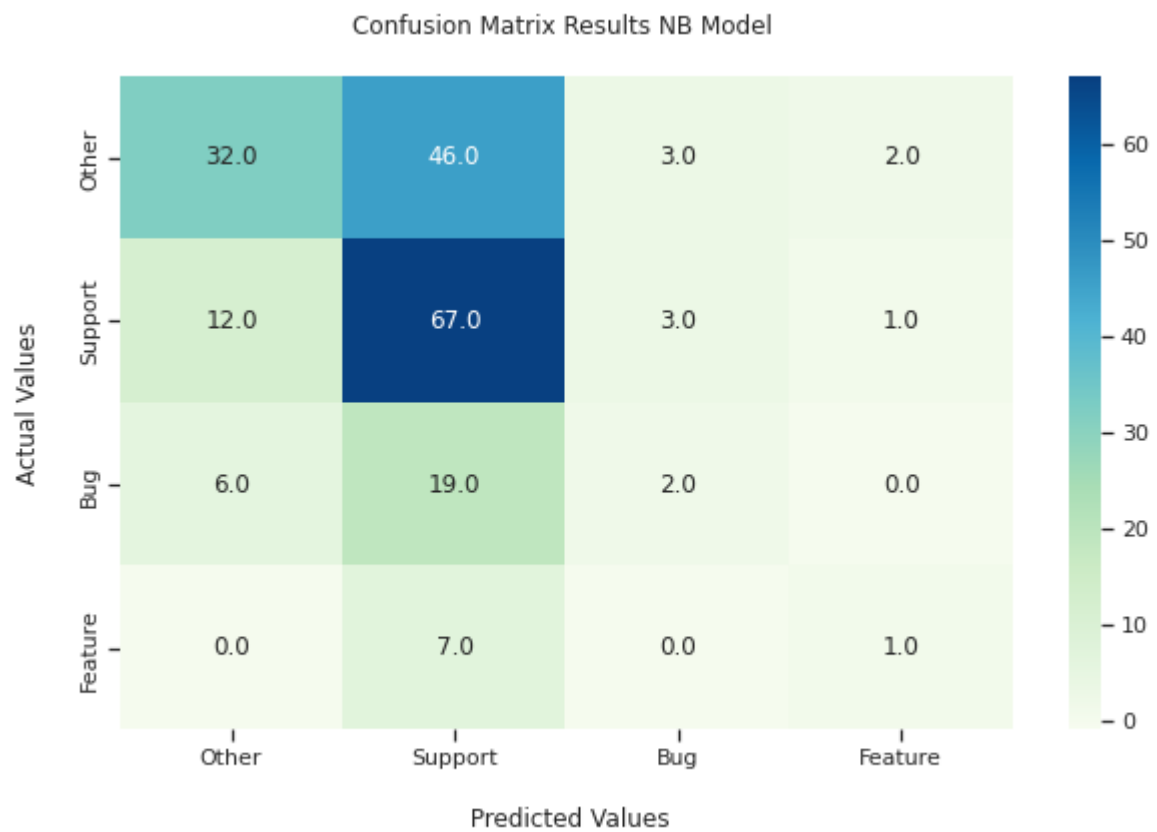
confusion_table_nb_SKL = t_labels.transform(confusion_table_nb_SKL)
confusion_table_nb_SKL = p_labels.transform(confusion_table_nb_SKL)

y_true = np.array(confusion_table_nb_SKL.select(col("TargetLabel")).collect())
y_pred = np.array(confusion_table_nb_SKL.select(col("PredictionLabel")).collect())

cm = confusion_matrix(y_true, y_pred, labels=["Other", "Support", "Bug", "Feature"])
print(np.sum(cm))
confusion_matrix_plotter(cm, "NB Model", nparray='y')
```

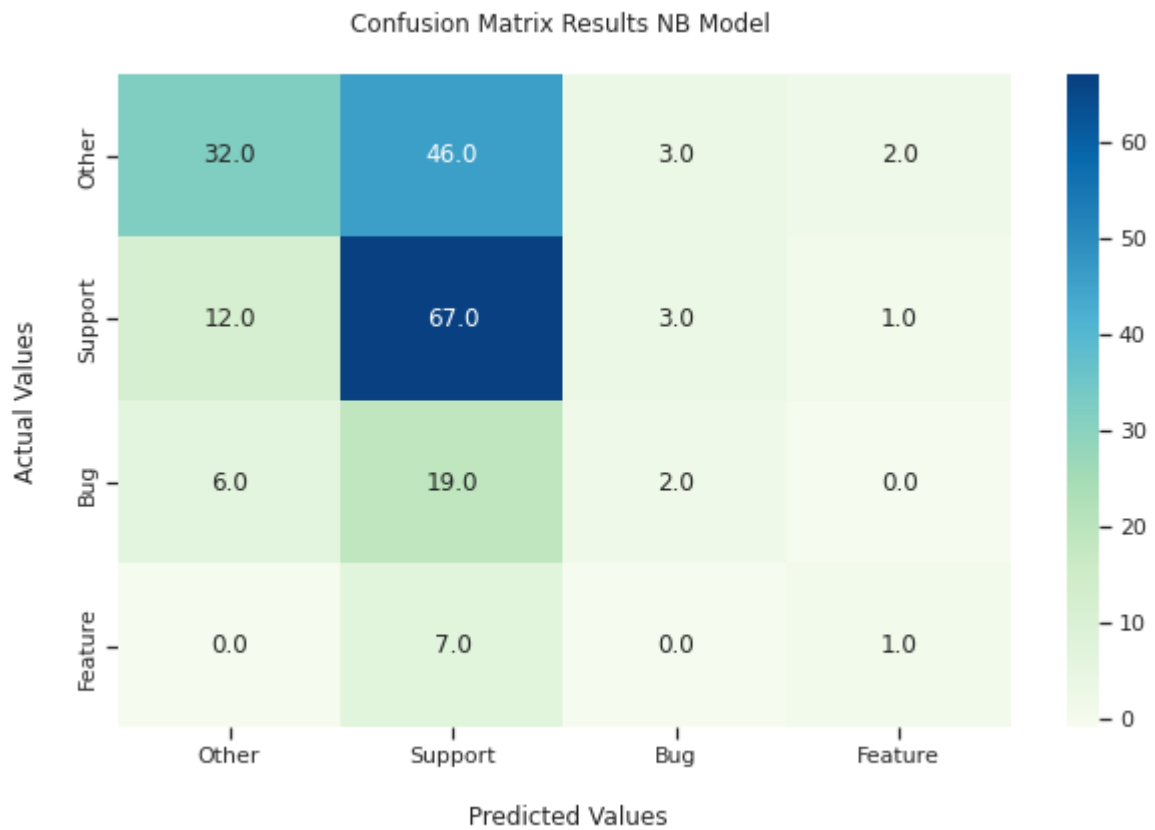
201

```
[[32 46  3  2]
 [12 67  3  1]
 [ 6 19  2  0]
 [ 0  7  0  1]]
```



```
In [ ]: # call helper to plot confusion matrix
confusion_matrix_plotter(predictions_best_model_nb, "NB Model")

[[32. 46.  3.  2.]
 [12. 67.  3.  1.]
 [ 6. 19.  2.  0.]
 [ 0.  7.  0.  1.]]
```



```
In [ ]: predications_and_labels = predictions_best_model_nb.select(col('label'),col('p
labels = ["Other", "Support", "Bug", "Feature"]
metrics = MulticlassMetrics(predications_and_labels.rdd)
```

```
In [ ]: print("Accuracy: ", metrics.accuracy)
for idx, label in enumerate(labels):
    print("Recall for {}: {}".format(label, metrics.precision(idx)))
    print("Precision for {}: {}".format(label, metrics.recall(idx)))
    print("f1 score for {}: {}".format(label, metrics.fMeasure(float(idx), 1.0)))
```

```
Accuracy: 0.5074626865671642
Recall for Other: 0.3855421686746988
Precision for Other: 0.64
f1 score for Other: 0.481203007518797
Recall for Support: 0.8072289156626506
Precision for Support: 0.48201438848920863
f1 score for Support: 0.6036036036036037
Recall for Bug: 0.07407407407407407
Precision for Bug: 0.25
f1 score for Bug: 0.11428571428571428
Recall for Feature: 0.125
Precision for Feature: 0.25
f1 score for Feature: 0.16666666666666666
```

Post-Analysis

Generate a list of all the misclassified issues for further manual analysis. We will use the best Logistic Regression classifier from the earlier grid search since it was our best overall model.

```

In [ ]: # get the predictions from the model for each sample
        final_predictions = predictions_best_model_lr.select(col('number'), col('prediction'))

        # # Get the labels back from the model
        from pyspark.ml.feature import IndexToString
        p_labels = IndexToString(inputCol='prediction', outputCol="PredictionLabel", labelCol='number')
        final_predictions = p_labels.transform(final_predictions)

        # join the predictions onto the original dataframe df, and filter for only misclassified
        misclassified = final_predictions.join(df, (final_predictions.number == df.number))
        # add all columns back from the original df by inner joining
        misclassified = misclassified.join(df, misclassified.number == df.number, "inner")

```

There are 95 total misclassified issues from the validation set.

```

In [ ]: misclassified.count()

```

Out[]: 95

Download the files saved by the steps above

```

In [ ]: # export the misclassified labels to csv and then download
        panda_df = misclassified.toPandas()
        panda_df.to_csv("Misclassified Issues.csv", index=False)
        files.download('/content/Misclassified Issues.csv')

```

```

In [ ]: #first zip the parquet files
        !zip -r /content/nlp.zip /content/nlp
        !zip -r /content/trainedmodels.zip /content/trainedmodels

```

```

In [ ]: #then download
        files.download('/content/nlp.zip')
        files.download('/content/trainedmodels.zip')

```


APPENDIX C – Jupyter Notebook
(prototype in sklearn and spaCy NLP)

Import the required packages and set up the Spark session.

```
In [ ]: # install required packages
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
%pip install -q pyspark
%pip install -q findspark
%pip install -q joblibspark
%pip install -q spacy
!python -m spacy download en_core_web_sm
```

```
In [ ]: import findspark
findspark.init()
findspark.find()
```

```
Out[ ]: '/usr/local/lib/python3.8/dist-packages/pyspark'
```

```
In [ ]: from pyspark.sql import SparkSession
import pyspark;
spark = SparkSession.builder.appName('612Project').getOrCreate();
```

Note: We will be using Spark for GridSearchCV only. This appears to be the only sklearn method currently supported by Spark. The remaining code will be executed using Pandas dataframes.

Load the CSV file (n=1000 samples) containing our manual labels as the target vector

```
In [ ]: # This notebook was originally run in Google colab due to better hardware perf
# This code should not be run locally (it will not work)

# from google.colab import files
# uploaded = files.upload()
```

No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving GH-React.csv to GH-React.csv

```
In [ ]: import pandas as pd

df = pd.read_csv("./GH-React.csv")

#keep only the columns we need
df = df[['title', 'author_association', 'body', 'Target']]
```

```
In [ ]: print("Shape:", df.shape)
df.columns
```

```
Shape: (1000, 4)
```

```
Out[ ]: Index(['title', 'author_association', 'body', 'Target'], dtype='object')
```

Machine Learning Pipeline

Stage 1

The preprocess() function is defined below. It takes in a String formatted as Markdown from GitHub and pre-processes it to return a new string ready for the next stages in our ML Pipeline.

```
In [ ]: import re

def preprocess(text):
    stripped = text.lower()

    # remove all headings, bold text, and HTML comments from the Markdown text
    # These items have all been used by the React team in their issue template.
    headings_pattern = r'(<=\\s|^){1,6}(\\.\\*?)$'
    bold_pattern = r'\\*\\*(\\.\\+?)\\*\\*(?!\\*)'
    comments_pattern = r'<!--(\\.|\\n)*?-->'
    combined_pattern = r'|'.join((headings_pattern, bold_pattern, comments_pattern))

    stripped = re.sub(combined_pattern, '', stripped)

    # find all URLs in the string, and then remove the final directory from each
    # there may be useful patterns based on what URLs issues are commonly linked to
    url_pattern = re.compile(r'(https?:\\/\\/[^\\s]+)')
    for url in re.findall(url_pattern, stripped):
        new_url = url.rsplit("/", 1)[0]
        stripped = stripped.replace(url, new_url)

    non_alpha_pattern = r'^A-Za-z ]+'
    stripped = re.sub(non_alpha_pattern, '', stripped)

    return ' '.join(stripped.split())
```

```
In [ ]: #convert body and title column to unicode, there were some issues with process.
df['body'] = df['body'].astype('U')
df['title'] = df['title'].astype('U')
```

Test the preprocess function on a sample post to ensure that it works as expected:

```
In [ ]: test = df['body'][4]
preprocess(test)
```

```
Out[ ]: 'bug or undefined behaviourdoingreactchildrentoarray reactdomcreateportal fails
withobjects are not valid as a react child found object with keys typeof key c
hildren containerinfo implementation if you meant to render a collection of ch
ildren use an array insteadnamely the following complete snippet failsjsximport
t react from reactimport render createportal from reactdomconst renderchildren
children children reactchildrentoarraychildren return hrenders children with t
oarray childrenhconst app renderchildren namecodesandbox createportaldivrender
ed in portaldiv documentgetelementbyidportal renderchildrenrenderapp documentg
etelementbyidrootwhile the following one which wraps the portal in another ele
ment works just finejsximport react from reactimport render createportal from
reactdomconst renderchildren children children reactchildrentoarraychildren re
turn hrenders children with toarray childrenhconst app renderchildren namecode
sandbox div createportaldivrendered in portaldiv documentgetelementbyidportal
div renderchildrenrenderapp documentgetelementbyidrooti am aware that createpo
rtal is a new feature but in the best case scenario it should be possible to u
se it everywhere other valid nodes are acceptedthe same thing is happening for
reactcloneelementreactdomcreateportal its probably weird to try and clone a po
rtal but maybe we should specify in the createportal documentation that it can
not be cloned at least for now should i open a pr for thatlet me know your tho
ughtsim using react'
```

Stage 2

Split the data into training (80%) and validation(20%) sets. We will stratify based on the label since our dataset is imbalanced.

```
In [ ]: y = df['Target']
X = df.drop(['Target'], axis=1)
```

```
In [ ]: from sklearn.model_selection import train_test_split

X_train, X_val, y_train, y_val = train_test_split(X, y, train_size=0.8, strati

print(X_train.shape)
print(X_val.shape)

(800, 3)
(200, 3)
```

Stage 3

Create a TF-IDF features matrix using TfidfVectorizer from sklearn applied to the title and body of each issue.

We will additionally add in the feature 'author_association' from the GitHub issue, as there may be a correlation between Members/Collaborators/Contributors submitting more valid bugs/feature requests than "None" users.

While lemmatization could have been done earlier in the pre-processsing stage, it is more efficient to lemmatize at this point in a custom_tokenizer() function passed to TfidfVectorizer since tokenization is part of both processses.

First, define the tokenizer and vectorizer:

```
In [ ]: import spacy
        from sklearn.feature_extraction.text import TfidfVectorizer

        nlp = spacy.load("en_core_web_sm")

        # create a custom tokenizer using the spacy document processing pipeline
        def custom_tokenizer(document):
            ppd = preprocess(document)
            doc = nlp(ppd)
            return [token.lemma_ for token in doc]

        tfidfvect = TfidfVectorizer(tokenizer=custom_tokenizer, ngram_range=(1, 2), mi
```

We will also use one-hot-encoding on the author-association feature

```
In [ ]: from sklearn.preprocessing import OneHotEncoder

        # use one hot encoder to transform the author_association to a feature set
        ohe = OneHotEncoder()
```

Create the features matrix using a ColumnTransformer to create a pipeline with the different feature generation methods. We will use a separate vectorizer on the body and title to produce a different set of features for each. The tokens in the title may hold different importance than the same token in the body.

```
In [ ]: from sklearn.compose import make_column_transformer
        from sklearn.compose import ColumnTransformer

        ct = ColumnTransformer(
            [("title", tfidfvect, "title"),
             ("body", tfidfvect, "body"),
             ("ohe", ohe, ['author_association'])])

        ct.fit(X_train)
        X_train_trans = ct.transform(X_train)
        X_train_trans.shape
```

```
Out[ ]: (800, 3544)
```

Perform an initial analysis on our model to see how our train and validation scores look (spoiler: not great)

```
In [ ]: from sklearn.linear_model import LogisticRegression
        logreg = LogisticRegression(max_iter=1000)
        logreg.fit(X_train_trans, y_train)

        X_val_trans = ct.transform(X_val)
        print("Train score: {:.2f}".format(logreg.score(X_train_trans, y_train)))
        print("Validation score: {:.2f}".format(logreg.score(X_val_trans, y_val)))
```

```
Train score: 0.89
Validation score: 0.57
```

Look at the features with the lowest and highest idf from the 'body' column just to see if things look reasonable. The features with the lowest idf are what we would think of as 'stop words', so this seems intuitive.

```
In [ ]: import numpy as np
sorted_by_idf = np.argsort(ct.named_transformers_.body.idf_)
feature_names = np.array(ct.named_transformers_.body.get_feature_names_out())

print("Features with lowest idf:\n{}".format(feature_names[sorted_by_idf[:20]])
print("Features with highest idf:\n{}".format(feature_names[sorted_by_idf[-20:]

Features with lowest idf:
['the' 'be' 'to' 'a' 'not' 'in' 'and' 'this' 'I' 'react' 'it' 'of' 'do'
 'use' 'version' 'that' 'for' 'with' 'component' 'have']
Features with highest idf:
['standalone' 'eject' 'start build' 'either the' 'start with' 'state I'
 'devdependencie' 'state dispatch' 'during the' 'down the' 'still be'
 'still have' 'do in' 'dispatch type' 'discussion' 'disabled' 'diff'
 'subcomponent' 'state for' 'your time']
```

TODO: See if we can visualize our text features to make sure it seems logical. Take an example from Chapter 7 in ML book.

Stage 4

Grid Search for optimizing model

Use grid search to find potentially better model parameters:

```
In [ ]: from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline
import numpy as np
from sklearn.model_selection import GridSearchCV

##### import joblibspark for gridsearch
from joblibspark import register_spark
from sklearn.utils import parallel_backend
register_spark() # register spark backend

pipe = make_pipeline(ct, LogisticRegression(max_iter=1000))

param_grid = {"logisticregression__C": [0.1, 0.5, 1, 2, 5],
              "columntransformer__body_ngram_range": [(1, 1), (1, 2), (1, 3)
              "columntransformer__body_min_df": [1,5]
              }

In [ ]: # WARNING: Running this cell will run GridSearch. Skip to loading the pickle f.

with parallel_backend('spark', n_jobs=-1):
    grid = GridSearchCV(pipe, param_grid, cv=5)
    grid.fit(X_train, y_train)
```

```
In [ ]: # write the grid object to a file so that it can be loaded in a different sess.
import dill as pickle
pickle.dump(grid, open("grid.pkl", "wb"))
```

```
In [ ]: # this was originally run on Google colab as their hardware is better than mine
# download the results from colab to my local drive
from google.colab import files
files.download('grid.pkl')
```

```
In [ ]: # load the grid variable back from file to continue using it's
# contents for analysis in future sessions
grid = pickle.load(open("grid.pkl", "rb"))
```

```
In [ ]: print("Best parameters: {}".format(grid.best_params_))
print("Best cross-validation score: {:.2f}".format(grid.best_score_))
```

```
Best parameters: {'columntransformer__body__min_df': 1, 'columntransformer__bo
dy__ngram_range': (1, 3), 'logisticregression__C': 0.5}
Best cross-validation score: 0.56
```

Stage 5

Use the best parameters generated by grid transform to re-train and validate our model with the X_val and y_val data we saved in stage 3:

```
In [ ]: tfidfvect = TfidfVectorizer(tokenizer=custom_tokenizer, ngram_range=(1, 3), min

ct = ColumnTransformer(
    [("title", tfidfvect, "title"),
     ("body", tfidfvect, "body"),
     ("ohe", ohe, ['author_association'])])

X_train_trans = ct.fit_transform(X_train)

logreg = LogisticRegression(C=0.5, max_iter=1000)
logreg.fit(X_train_trans, y_train)

X_val_trans = ct.transform(X_val)
print("Train score: {:.2f}".format(logreg.score(X_train_trans, y_train)))
print("Validation score: {:.2f}".format(logreg.score(X_val_trans, y_val)))
```

```
Train score: 0.87
Validation score: 0.55
```

Our model doesn't perform great, but at least it beats random chance by about 20%!

```
In [ ]: from sklearn.dummy import DummyClassifier
dummy_clf = DummyClassifier(strategy="stratified", random_state=0)
dummy_clf.fit(X, y)

print("Score based on chance: {:.2f}".format(dummy_clf.score(X, y)))
```

```
Score based on chance: 0.35
```