**Github bug prediction using Random Forest Classifier using R**

**Summary:** Using GitHub to anticipate bugs, features, and questions might be advantageous for better resource use.  The GitHub Bugs Prediction dataset from Kaggle is utilized for forecasting, and Random Forest Classification using TF-IDF is employed to predict bugs, features, and questions based on GitHub titles and text content. This report will compare the Random Forest performance evaluations for different tree counts. With such a massive dataset comprising text data, there is a lot to consider while analyzing it, primarily because of the preprocessing required to represent raw text and make it adaptable for machine. This project followed multiple steps which are mentioned below:

1. Retrieving data from json dataset and converting the data into data frames
2. Data cleaning:

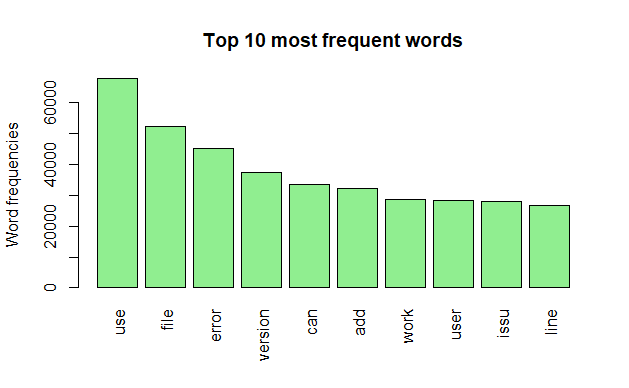
* Removing http links
* Removing html tags
* Removing user names
* Removing unnecessary English stop-words
* Removing single letters & Punctuation
* Removing additional white spaces
* Removing numbers

1. Data Preprocessing:

* Making text lower cases
* Stemming: reducing words to their stem

1. Creating Document Term Matrices (DTM) with an implementation of the Bag of Words
2. Reducing sparsity
3. Splitting data (80:20) for training & testing
4. Implementing and tuning Random Forest Classification
5. Demonstration of feature importance along with model evaluation

After cleaning the data and pre-processing, Document Term Matrices are created using the "bag-of-words" model, in which the appearance of each word is utilized to train a classifier. Later, sparsity (matrices with a large number of zero values) is reduced, the data is splitted into 80:20 for training and testing followed by implementing machine learning model. Based on annotated texts (Training Data), a ML model can be trained for prediction. In this project, Random Forest Classifier is used. To generate a more precise and reliable prediction, Random Forest creates many decision trees and merges them together. The random forest approach has two crucial parameters: the number of trees in the forest (ntree) and the number of random variables utilized in each tree (ntree). The default value for mtry is 8 (number of variables tried at each split). The random forest with multiple ntree values (100, 200, and 300) is generated to determine the number of trees that correlate to a stable classifier. After measuring the accuracy, Kappa, and OOB (Out of Bag) error rates for each ntree value and comparing the number of trees where the rates settle, ntree= 300 is picked as the top performer.

****

|  |  |
| --- | --- |
| **Term** | **Frequency** |
| use | 67570 |
| file | 51994 |
| error | 45143 |
| version | 37429 |
| can | 33510 |
| add | 32133 |
| work | 28526 |
| user | 28338 |
| issu | 27784 |
| line | 26697 |

Figure 1: Most frequent words

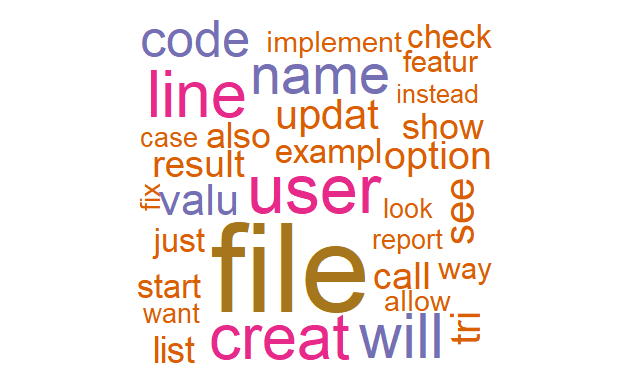
****

Figure 2: Word cloud of most frequent terms

Term frequency is calculated on corpus (a collection of text). Document Term Matrices are created using the "bag-of-words" model, in which the appearance of each word is utilized to train a classifier. So, in a nutshell, it's a tally of all the words in the document. It consists of a lexicon of known terms as well as an indicator of their prevalence. The Term Document Matrix keeps track of how often each term appears in each document. With a Bag of Words depiction of the documents, then track the number of times each term appears in each document. Depending on the number of documents in the corpus and the number of terms in each document, a Document Term matrix can become a very big, sparse matrix (with more 0s than values). Hence reducing the sparse is quite important as well.

**Feature importance of Random Forest Classification:**

The importance of features can be estimated from data by building a model. The random forest approach has the advantage of making determining the relative importance of each feature on the prediction standardized. After training, it calculates this score for each feature and weighs the findings so that the total importance equals one. It is feasible to pick which features to eliminate based on their importance because they do not contribute enough to the prediction process. With more features, the model is much more likely to experience overfitting. The Random Forest method includes a feature importance calculation that can be conducted in two manners:

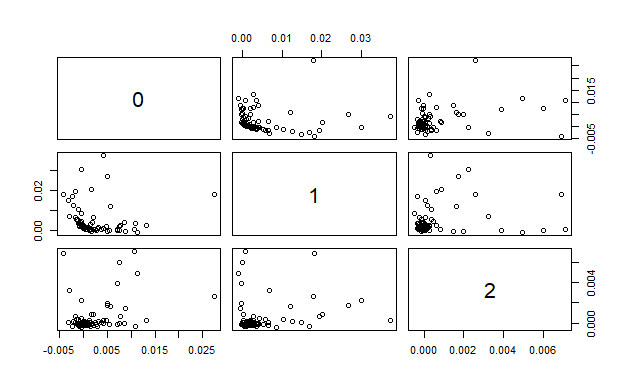


Figure 3: Plotting for feature importance of RF model

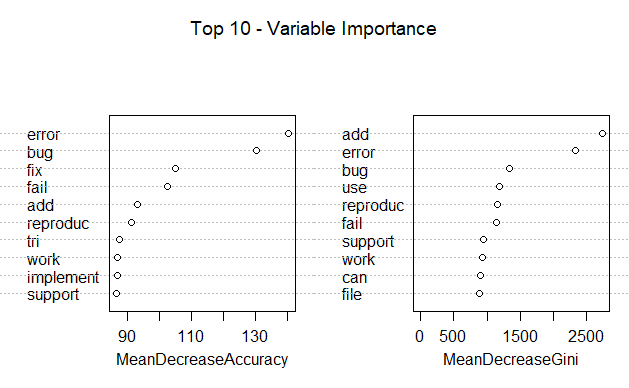
****

Figure 4: Top 10 most important features

**Mean Decrease Gini**: A Random Forest is a collection of Decision Trees, each of which is made up of internal nodes and leaves. Internal node characteristics are chosen using a condition, which in classification problems can be Gini impurity. It can be seen how each feature reduces the split's impurity (the feature with the highest decrease is selected for internal node). It can be calculated how each feature reduces impurity on average for each feature. The feature importance is calculated as the average of all trees in the forest.

**Mean Decrease Accuracy**: It is a method for calculating the relevance of features on permuted out-of-bag (OOB) samples depending on a mean decrease in accuracy.

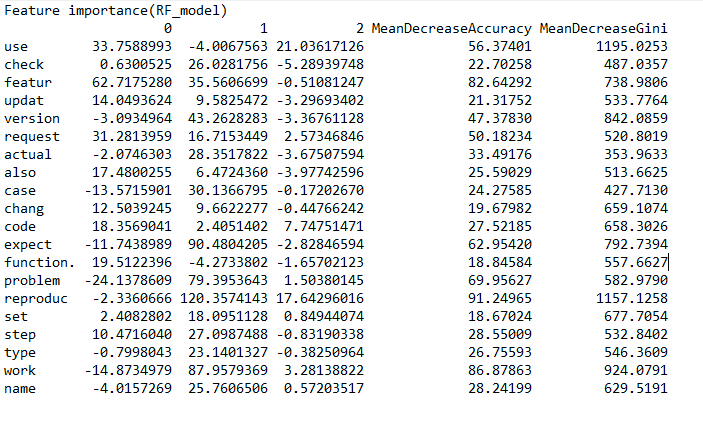
****

Figure 5: Feature Importance of RF model (20 out of total)

**Result Analysis with Performance metrics**

Evaluation parameters are used to compare the results of models implemented on the dataset. There are several performance metrics as evaluation parameters for the detection system. The performance of the bug prediction is assessed using the following criteria: accuracy, classification error, F1 score, confusion matrix, and so on. The out-of-bag (OOB) error is a way of evaluating the accuracy of a random forest and selecting optimal values for tuning parameters like mtry. A brief description of each performance metrics to evaluate the Random Forest Classification model for bug prediction are shown below:

* 1. Accuracy (Acc): Percentage of correctly identified for prediction.

Acc=(TP+TN)/(TP+TN+FP+FN)

* 1. Classification Error (Err): Percentage of incorrectly identified for prediction.

Err= (FP+FN)/(TP+TN+FP+FN)

* 1. Precision (positive predictive value): is the fraction of relevant instances among the retrieved instances

Precision=TP/ (TP+TP)

* 1. Recall (sensitivity): is the fraction of relevant instances that were retrieved.

Recall=TP/ (TP+FN)

* 1. F1 score: It conveys the balance between the precision and the recall. A high F1 score indicates that serious threats are correctly identified and that false alarms are avoided.

F1 Score= 2\*((precision\*recall)/(precision+recall))

* 1. Kappa: (Observed Accuracy – Expected Accuracy) / (1 – Expected Accuracy); it takes into account the accuracy that would have happened anyway through random predictions.
  2. Detection Prevalence: It shows the number of positive class predictions made as a proportion of all predictions.
  3. Balanced accuracy: is calculated as the average of the proportion corrects of each class individually.
  4. Confusion Matrix:While the accuracy measuring metrics provide a general overview, the confusion matrix shows the prediction's actual results. It divides the results into four different categories, True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN). Generating a confusion matrix by plotting with the package becomes more comfortable to visualize.
  5. AUC (Area under ROC curve)**:** This is another performance measurement metric for classifiers. ROC (Receiver Operating Characteristics) curve is a probability curve that distinguishes between the true positive rate and false positive rate. AUC determines the fraction of area that falls underneath the ROC curve.

**Comparison of Accuracy due to the changes of trees by tuning RF model manually:**

|  |  |  |
| --- | --- | --- |
| **Number of trees** | **Accuracy** | **Kappa** |
| 100 | 87.96% | 0.7890 |
| 200 | 87.99% | 0.7896 |
| 300  (Chosen ntree for the classification) | 88.06% | 0.7909 |

**Overall Statistics:**

|  |  |
| --- | --- |
| Classification Accuracy | 0.8806 |
| 95% CI (confidence interval) | (0.8765, 0.8847) |
| No Information Rate | 0.5098 |
| P-Value [Acc > NIR] | < 2.2e-16 |
| Kappa | 0.7899 |
| Mcnemar's Test P-Value | < 2.2e-16 |

**Out of Bag (OOB) Error:**

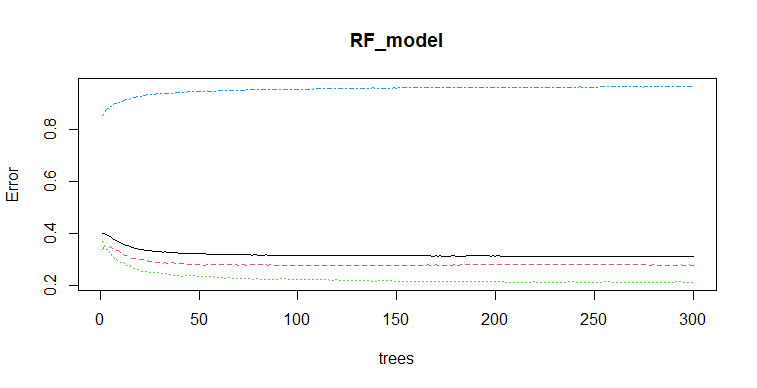
****

Figure 6: Out of Bag error

**No of Nodes for the trees:**

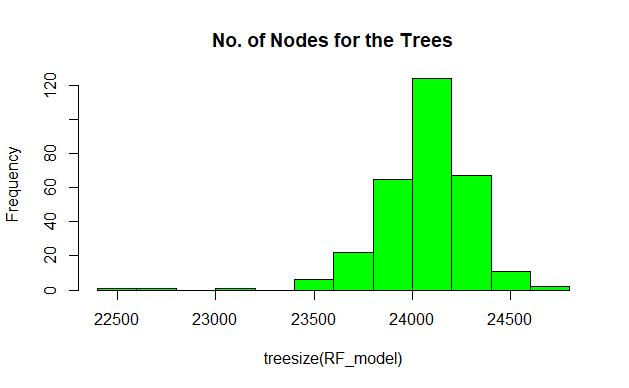
****

Figure 7: No of nodes for the trees

Random forest is a beneficial approach since it usually produces reasonably good results with the default hyperparameters (ntree & mtry).  Even with more features, the classifier is unlikely to overfit the model if there are enough trees in the forest. The biggest drawback of random forest is that it can become too sluggish and ineffective for real-time forecasts if there are too many trees.

**Confusion Matrix and Statistics (predictRF):**

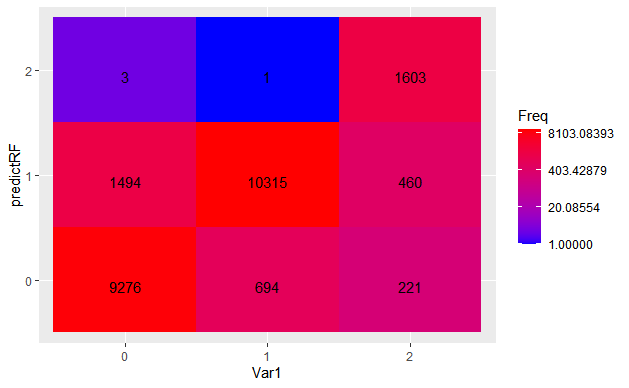


Figure 8:Confusion matrix of Random Forest Classification

|  |  |  |  |
| --- | --- | --- | --- |
| Row/Column | **0** | **1** | **2** |
| **0** | 9276 | 1494 | 3 |
| **1** | 694 | 10315 | 0 |
| **2** | 221 | 460 | 1603 |

**Statistics by Class:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Class: 0** | **Class: 1** | **Class: 2** |
| Sensitivity/ Recall | 0.9102 | 0.8407 | 0.99751 |
| Specificity | 0.8921 | 0.9411 | 0.96968 |
| Positive Predictive Value/ Precision | 0.8610 | 0.9369 | 0.70184 |
| F1 Score | 0.8849 | 0.8861 | 0.8239 |
| Neg Predictive Value | 0.9312 | 0.8503 | 0.99982 |
| Prevalence | 0.4234 | 0.5098 | 0.06677 |
| Detection Rate | 0.3854 | 0.4286 | 0.06661 |
| Detection Prevalence | 0.4476 | 0.4575 | 0.09490 |
| Balanced Accuracy | 0.9012 | 0.8909 | 0.98360 |

**AUC score & ROC Curve:**

Area under curve of random forest: 0.94139

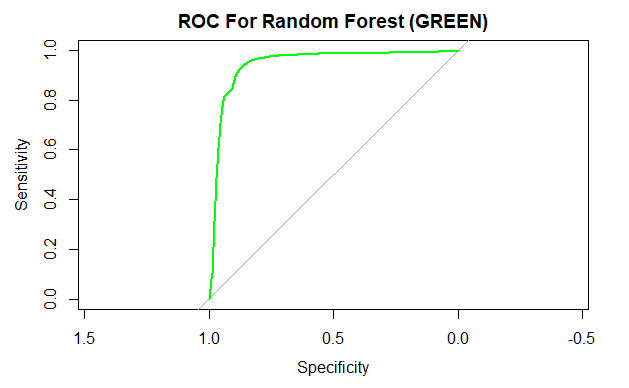


Figure 9: ROC curve for Random Forest Model