# **Classification of Bugs Using Machine Learning**

**Abstract :** This project is focused on solving the problem of classification of bugs in verification regressions using the supervised machine learning techniques. Classifying the bugs to one of the categories respectively by manually is time taking and this project helps to classify them automatically without manual effort.

**Introduction :**

The goal of this project is to classify the bugs in to different categories of Major Blocks, Sub Blocks and Sub Sub-Blocks using Machine Learning Algorithms.

First we will classify the bugs in to major blocks based on the data, and then classify the sub blocks based on major blocks and sub sub blocks based on sub blocks. This project completely follows the life cycle of a data science project starting from Data collection to Model evaluation and retraining approach except the deployment.

We have used Random Forest Classifier to classify the bugs in the categories of major blocks. Up on that Hyper parameter tuning, Synthetic Minority Over Sampling Technique (SMOTE) and Random Over Sampler has been used to increase the performance of model and make the dataset to balanced.

Each models evaluation metrics and performance have been plotted visually using the seaborn and matplotlib.

**Contents Page**

1. Data Collection 3
2. Data Preprocessing 3
3. Data Visualization 3-9
4. Feature Engineering 9
5. Feature Selection 10
6. Data Standardization 10
7. Model Training 10-11
8. Model Evaluation Metrics 12-18
9. Hyper-Parameter Tuning 19-22
10. Visualizing Model Performance 12-22
11. Model Inferencing 22
12. Conclusion 23

**Data Collection :**  The dataset is collected from the simscope tool where all the verification regressions runs and it will populate the bugs in those regressions and those bugs will be assigned to an engineer to get it resolved. From that reported bugs in a specific timline we have collected the data. Simscope tool provides the opportunity to download the data in certain formats like xml, word and csv. To understand the data and solve the problem using ML we thought it is better to proceed with csv format. The data is collected in the format of csv, it has 1000 records and more than 500 columns.

**Data Preprocessing :** The collected data has many columns and there is a one columnIn this stage we have considered the records that containing the Bug in “Issue type” column and avoided the rest of the records.

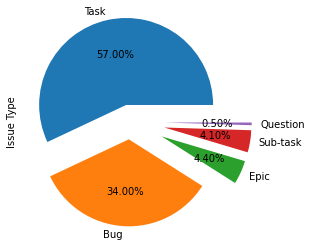
Data Preprocessing is the stage where we process the data to make ensure that it will fit the model, which includes handling the null values, finding categorical and numerical features, handling categorical features, reading the statistical information, determining and handling outliers, determining the correlation among the features.

All the features in dataset are of object type and it has no missing values.

**Data Visualization :** Data visualization is the process of visualizing the preprocessed data / original data with the help of pie chart, bar graph, line plot ,scatter plots and many more.

For visualization we have used matplotlib and seaborn libraries.

* ***visualizing the number of records in Issue type***



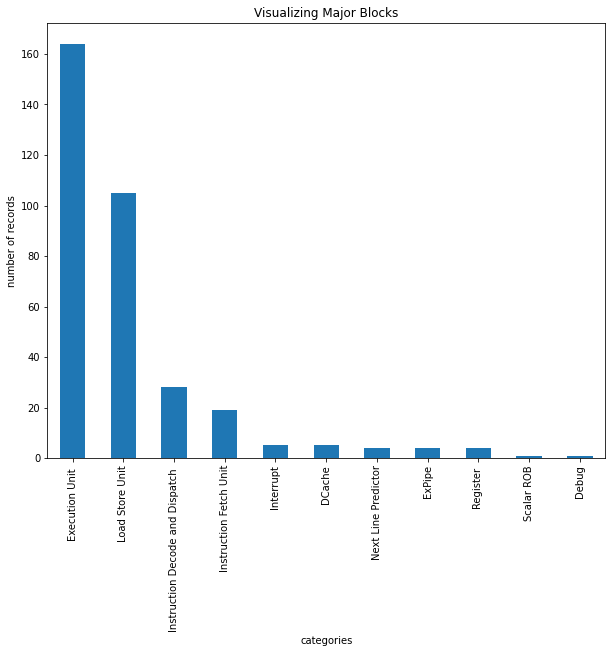
The above pie chart shows the percentage of records that belongs to categories of “Issue Type” column. We are considering only “Bug” type in this project, so we have 340 records in our dataset (34% are of Bug type out of 1000 records).

* ***visualizing the number of records in major blocks***

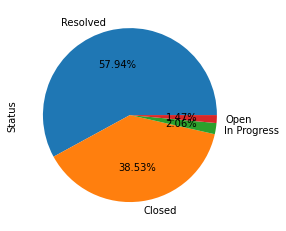
**We have 11 categories in Major Blocks column which is our target/output feature. Output of the model has to predict one of these 11 categories.**

**The number of records for each categories in major blocks have been visualized with the help of bar graphs using matplotlib library.**

**The categories like “Execution Unit” and “Load Store Unit” have more than 100 records each which is much more than rest of the categories and which makes our dataset as imbalanced.**

******

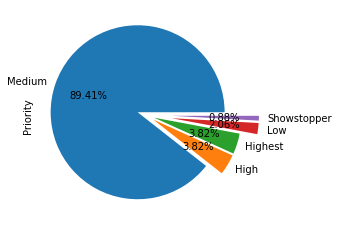
* ***visualizing the number of records in Status***



There are 5 categories in “Status” feature and we have visualized the percentage of records that belongs to each category with the help of pie charts using matplotlib library.

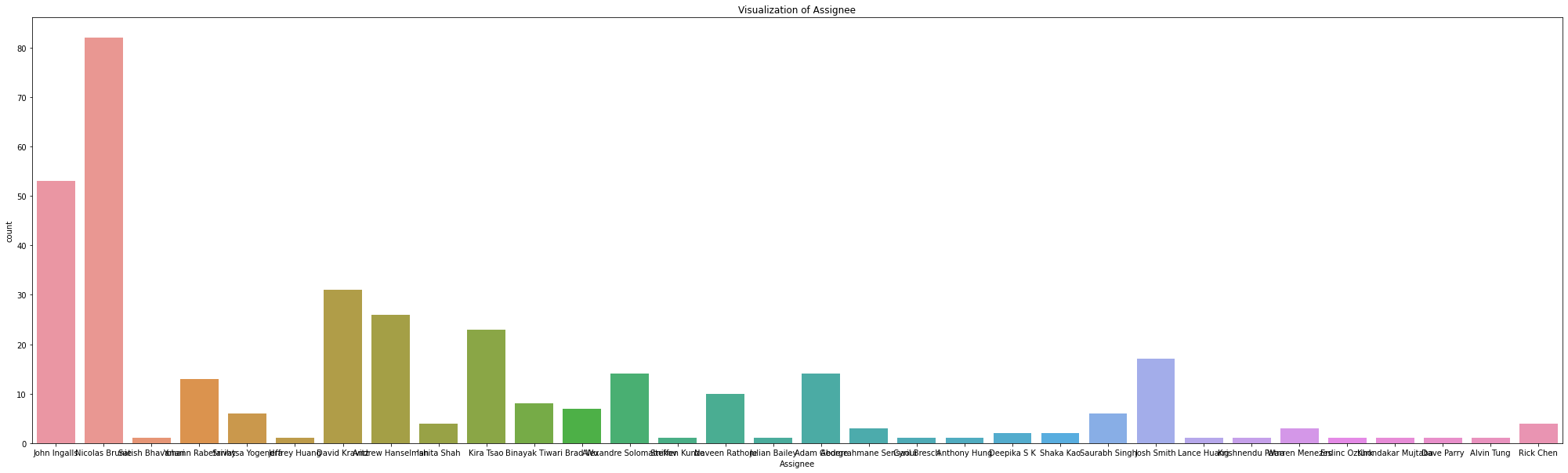
* ***visualizing the number of records in Priority***

There are 5 categories in “Priority” feature and we have visualized the percentage of records that belong to each category with the help of pie charts using the matplotlib library.



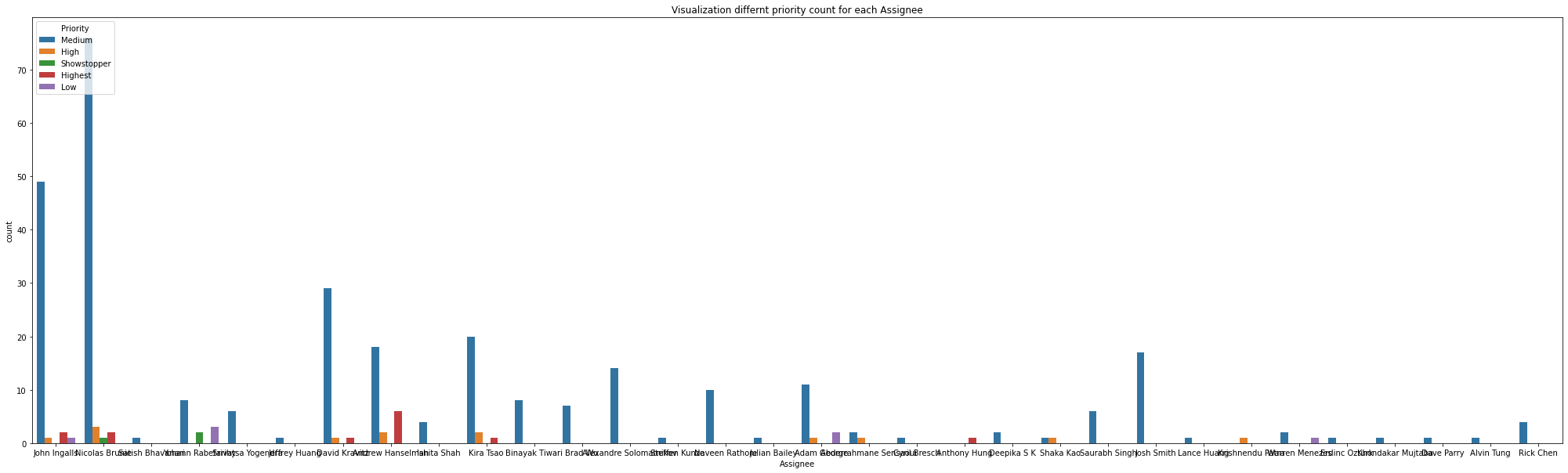
* ***Visualizing the Assignee column***

**There are 32 categories in “Assignee” column, all these are names of the assignee for whom the bug was assigned. Bar graph is used to visualize this feature which tells the number of records that belongs to each Assignee.**

******

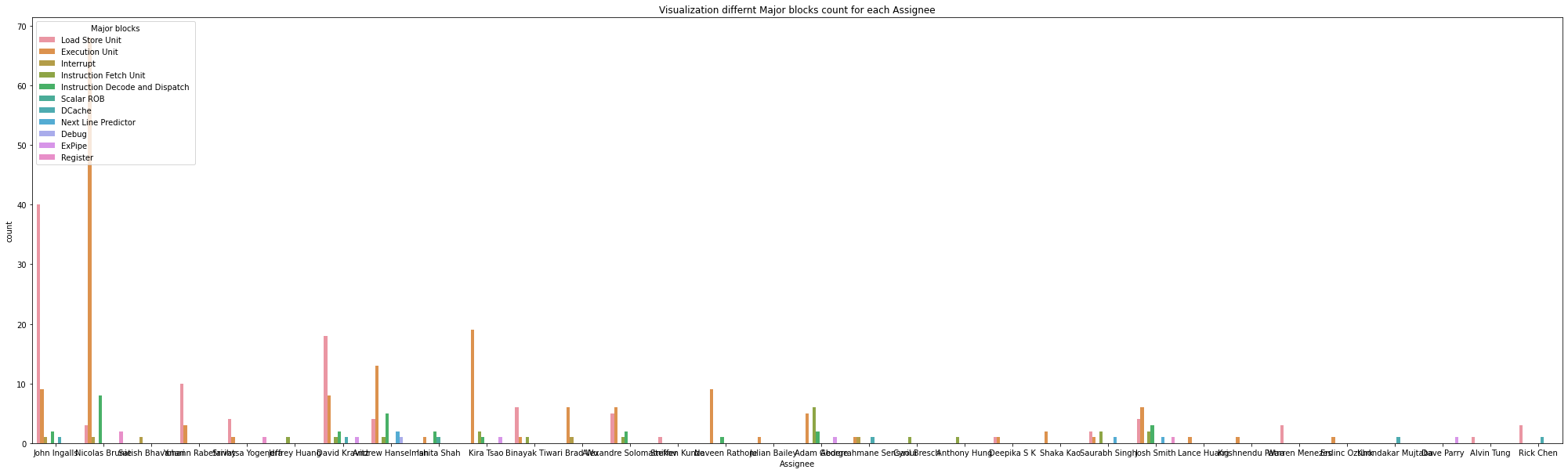
* ***Visualizing the differnt priority count for each Assignee***

**For each Assignee the bug assigned has a different priority. We are visualizing the number of records for each Assignee based on different priority with the help of countplot in seaborn library.**

******

* ***Visualizing the differnt Major blocks count for each Assignee***

**For each Assignee the bug assigned has different major block. We are visualizing the number of records for each Assignee based of different majority block with the help of countplot using seaborn library.**



**Feature Engineering :**

Feature Engineering in machine learning is all about handling the categorical features which are not numerical (String/object type). Machine learning model accepts only the features/columns that are numerical, to make these categorical features to numerical, there are certain feature engineering techniques which will convert these features to numeric.

* We have used Label Encoding for “Status” and “Priority” column.
* We have used one hot encoding for “Assignee” column.
* Extracted Date and Time from Created column and stored them in Date and Created\_Time column.
* Extracted Day and Month from Date column.
* Filtered punctuations, special characters and numbers from Summary column using regex.
* Implemented the concepts of NLP like Stopwords, Lemmatization on Summary column.
* Converted the Summary to TF-IDF vector using scikit learn.

**Feature Selection :** We have selected the features based on domain experts, the selected features are

* Summary
* Issue Type
* Status
* Assignee
* Priority
* Created
* Major Blocks

while training the model, Issue Type column is dropped because we have considered only Bug type in it, i.e, all the records are of bug type.

**Data Standardization :** All the features in this dataset are categorical and we have done feature engineering to make them numerical. Now all the features are numerical and they are not on the same scale.

To bring down all the features on to same scale we have to do Standardization, i.e. Mean=0 and SD=1.

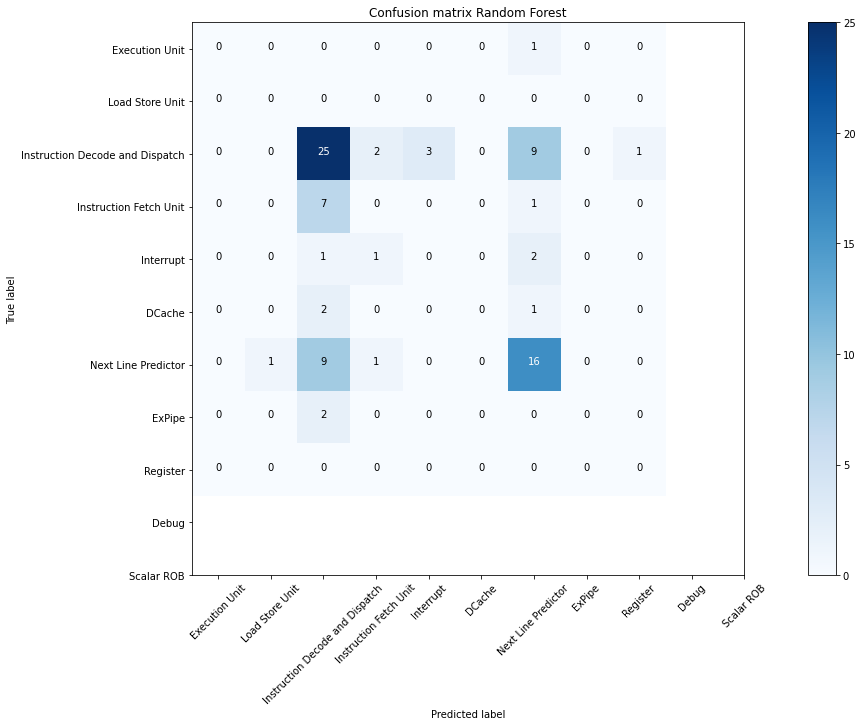
**Model Training :** For model training the entire dataset has to be split in to train and test, we have considered 80% for train and 20 % for test.

* ***Random Forest Model Performance***

Random Forest is supervised machine learning technique which is used to solve both regression and classification problem. Random Forest model built on decision trees and contain bunch of decision trees.

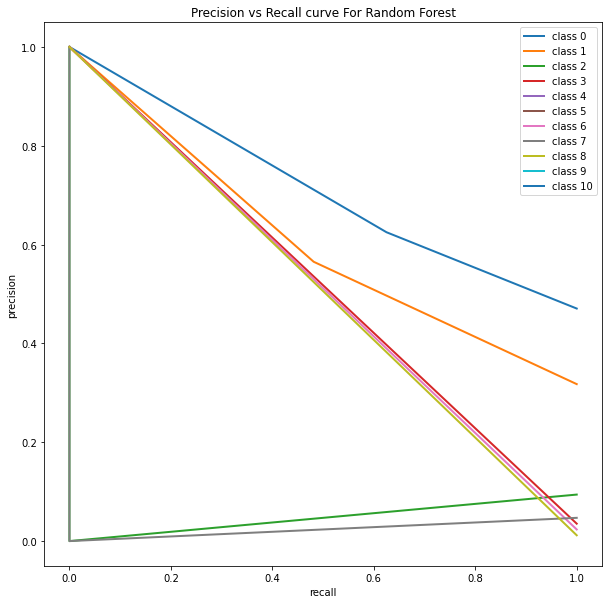
Accuracy 48.2%

**Confusion Matrix of Random Forest**



Confusion matrix help us how well our classifier has classified the output. With the help of confusion matrix we will evaluate our model with different metrics like precision, recall, f1score, accuracy and roc-auc score.

**Precision-Recall curve for Random Forest**



The precision-recall curve **shows the tradeoff between precision and recall for different threshold**. A high area under the curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate.

The above precision recall curve only holds good for 2 categories i.e. the area under the curve is maximum.

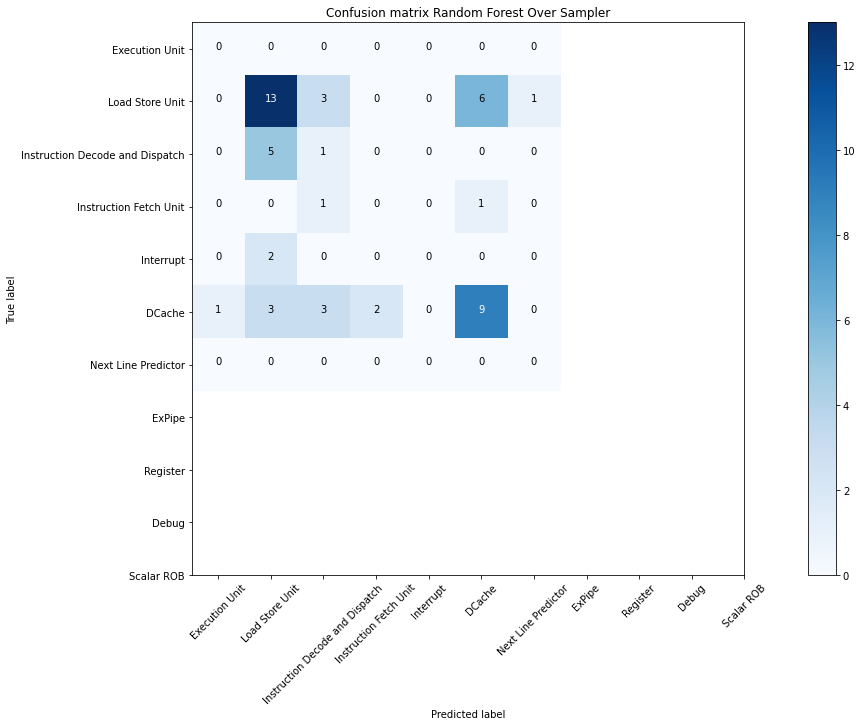
* ***Random Over Sampler on Random Forest Model Performance***

Accuracy 45%

We add synthetic rows to the data with the RandomOverSampler. We make the number of target values equal by increasing the minority class. This can be beneficial or harmful, depends on the quantities. However, we do not want to make our data synthetic.

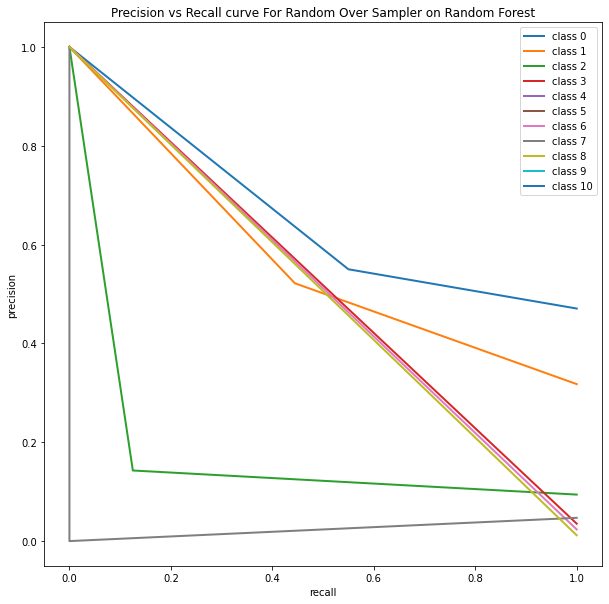
**Confusion Matrix of Random Over Sampler**

Confusion matrix help us how well our classifier has classified the output. With the help of confusion matrix we will evaluate our model with different metrics like precision, recall, f1score, accuracy and roc-auc score.



**Precision-Recall curve for Random over sampler on Random Forest**

The below shown precision recall curve only holds good for one category that is also just above the 50% of area under curve.



* ***SMOTE on Random Forest Model Performance***

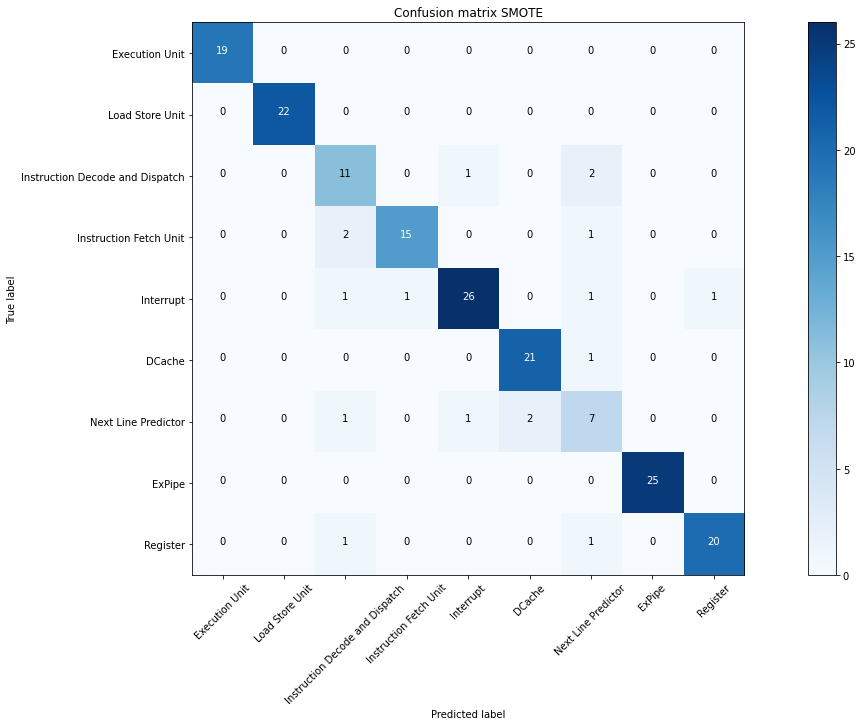
Accuracy 92%

SMOTE stands for Synthetic Minority Oversampling Technique.

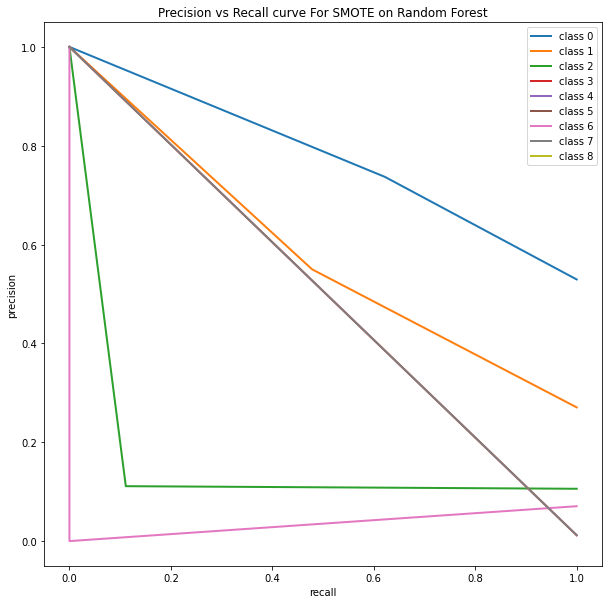
SMOTE is **an oversampling technique that generates synthetic samples from the minority class**. It is used to obtain a synthetically class-balanced or nearly class-balanced training set, which is then used to train the classifier.

SMOTE first selects a minority class instance a at random and finds its k nearest minority class neighbors. The synthetic instance is then created by choosing one of the k nearest neighbors b at random and connecting a and b to form a line segment in the feature space. The synthetic instances are generated as a convex combination of the two chosen instances a and b*.*

**Confusion Matrix for SMOTE on Random Forest**



**Precision-Recall curve for SMOTE on Random Forest**



The above shown precision recall curve also holds good for one category.

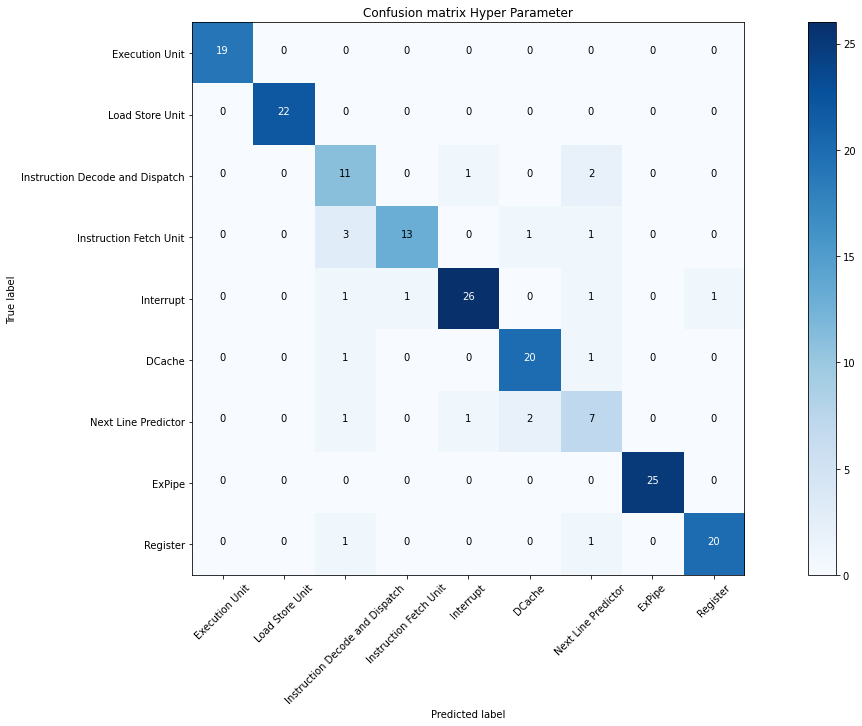
* ***Hyperparameter tuning on Random Forest***

Accuracy 89%

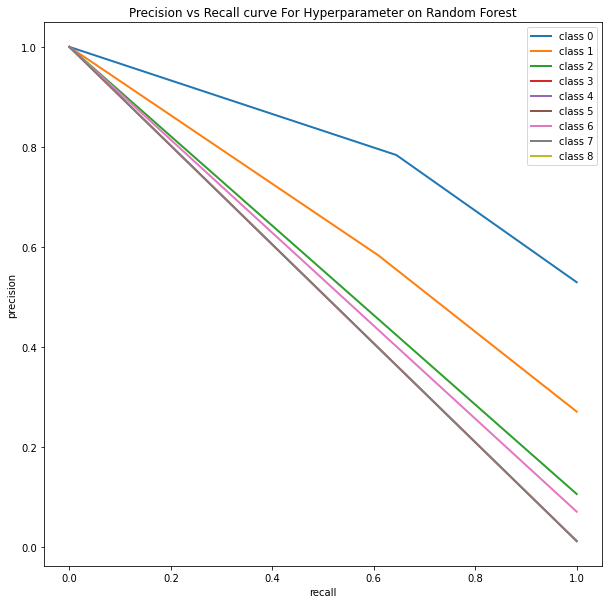
Hyperparameter tuning is choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a model argument whose value is set before the learning process begins. The key to machine learning algorithms is hyperparameter tuning.

For hyperparameter tuning we have used RandomizedSearchCV from scikit learn library.

**Confusion Matrix for Random Forest with Hyperparameter tuning**

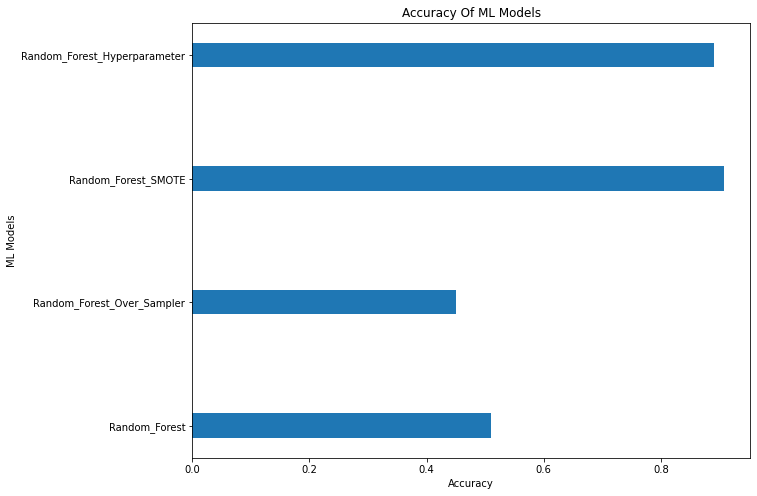


**Precision-Recall curve for Hyperparameter on Random Forest**



The above shown precision recall curve is much better than previous curves what we have seen. This is because we have used hyperparameter tuning with crossvalidation.

All Models Accuracy Score comparison



We have tried the four models to train our dataset out of those “Random Forest SMOTE with Hyperparameter” and “Random Forest SMOTE” has given accuracy around 90% and 92% respectively.

**Model Inferencing**

After training the model, we have to save the model for inferencing i.e. to predict the new samples.

Pickle library is used to save and load the model to reuse.

**Conclusion and Future Scope:**

This project was built on limited amount of data i.e. we have less number of records, because of that certain evaluation metrics of machine learning model will have lower performance. The performance this project in future will increase will increase definitely if we have huge amount of data and minimum/balanced records for each category in target column.