PROJECT REPORT
CMPE 255

PREDICTING CONTRIBUTING FACTORS FOR ROAD ACCIDENTS IN CHICAGO

Under the guidance of
Professor David Anastasiu

Date of Submission
05th December 2018

Team Members
Chaithra Lakshmi Sathyanarayana
Sayali Pisal
Thaijasa Badrinath Vijendranath

CHAPTER-1

Introduction

Many road safety measures have been implemented to decrease the number of traffic accidents and casualties.Wikipedia says that in the past 10 years, there have have been more than 300 million accidents every year. This has resulted in more than 35,000 deaths every year. This includes in-vehicle and pedestrian casualties. The statistics say that, some of the African countries have the highest number of accidents and casualties. European countries like Sweden have the lowest number accidents and casualties. Analysis of the contributory factors can be used by different countries, and implement measures to reduce the accidents. Traffic crashes and casualties have increased over time, due to increase in the number of vehicles. Accidents can have several contributing factors. Few of them are weather conditions, road surface conditions, rush-hour traffic, road construction, vehicle conditions, behavior of drivers. Analysis and evaluation of traffic accidents will enable the government to implement better traffic regulations and enhance road safety measures. The traffic crash data published by City of Chicago is used to train and test different classification models.

Motivation

A model to predict the primary contributory factors, based on the details available on the accident can be used to automate the process of prediction. This model can also be used to verify the primary contributory factor determined the traffic police, so as to reduce human error in predicting the primary contributory factor. Analysis of these factors can help the policy makers to implement new policies, and the city to improve infrastructure. This can also be used to make the general public aware of the different contributory factors, so that they can be alert of these factors, and avoid accidents.

Objective

The main objective of this project is to develop a model that can train and predict the primary contributing factors for traffic accidents. In this project, several feature selection techniques and supervised classification algorithms have been used to predict contributory factors, and the results have been evaluated using classification metrics. Comparing and contrasting the results of different algorithms is done to design a model that can accurately predict the contributory factors.

Literature Review

Several studies have conducted to analyze and predict causes for accident. Most of the studies for focus on predicting the cause based on specific factors, like weather conditions, work conditions(Poojitha Shetty et al., 2017; DurunDelen et al 2017., Yulan et al.,). In this project, we are considering all factors like external conditions, weather condition, driver behavior, were used to predict the primary contributory factors.

CHAPTER - 2

Algorithms Selection

1) AdaBoost Classifier:

Adaboost classifier combines several weak classifiers to provide a relatively accurate model. It works well even without tweaking much of the parameters. It helps in controlling overfitting the model.

2) Multi-Layer Perceptron:

Multi-layer Perceptron can have one or more hidden layers between input and output layers and it trains data using backpropagation. The model handles multi class classification by assigning probability method in the output function also called as SoftMax.

3) Bagging Classifiers:

This is an ensemble method which selects random subsets of data with replacement and fits base estimator with any of the other classifiers. Bagging classifier here is used with the following base estimators.

K Nearest Neighbors, Decision Trees Classifier, Extra Trees Classifier, Linear Discriminant Analysis Classifier and Random Forest Classifier.

4) Gaussian Naïve Bayes :

A conditional probability model used for managing continuous data values. The distribution of the continuous values associated with each class are done according to the Gaussian Normal distribution. Its main usage was to approximate the large variety of distributions in our dataset.

5) Logistic Regression:

A predictive analysis model which describes the data and draws inference relationship analysis for the binary attributes which depend on one or more nominal, ordinal, interval or ratio-level independent attributes. Implementation of this algorithm was done due to its property of handling multivariate dependent classes.

6) Random Forest:

Random Forest is a multi-analysis estimator computing different versions of decision tree classifiers on various data sub-samples. Sub-samples are chosen with replacement from the original dataset and measures to be of same size. A flexible and easy to use algorithm which provides averaging to improve the predictive accuracy and control over-fitting.

7) Extra Trees Classifier:

This algorithm differs from Random Forest where the best split is determined at each node from k random splits and each split is done without replacement.

8) Decision Tree Classifier:

A classification and regression supervised learning method which uses simple decision rules to infer the final outcome from the data attributes. Nonlinear relationship between parameters do not affect the tree performance as the tree implicitly performs variable screening and feature selection.

9) Linear Discriminant Analysis :
LDA uses Bayes rule for calculating statistical properties of data to draw linear decision boundary and make predictions. LDA performs linear combination of variables which best explain the data and does modelling to compute the difference between classes of data.

Tools and Technologies

Tools	Version	Application
Anaconda	4.5.11	An open source distribution of the python programming language which we used for applications like data processing, predictive analysis and scientific computations which helped to simplify package management and deployment.
HPC		System used for High Performance Computing (HPC). It provided powerful multi-core and multi-socket servers, high performance storage, GPUs, and large amounts of memory, tied together by an ultra-fast inter-connection network which we used for the computation of memory intensive algorithms like KNN.
Jupyter Notebook	5.7.1	Web application we used to create and share documents containing live code for pre-processing, model building, equations, visualisations and descriptive textual comments. Mainly used for : data cleaning and transformation, numerical simulation, statistical modelling and data visualisation.

Project Workflow:

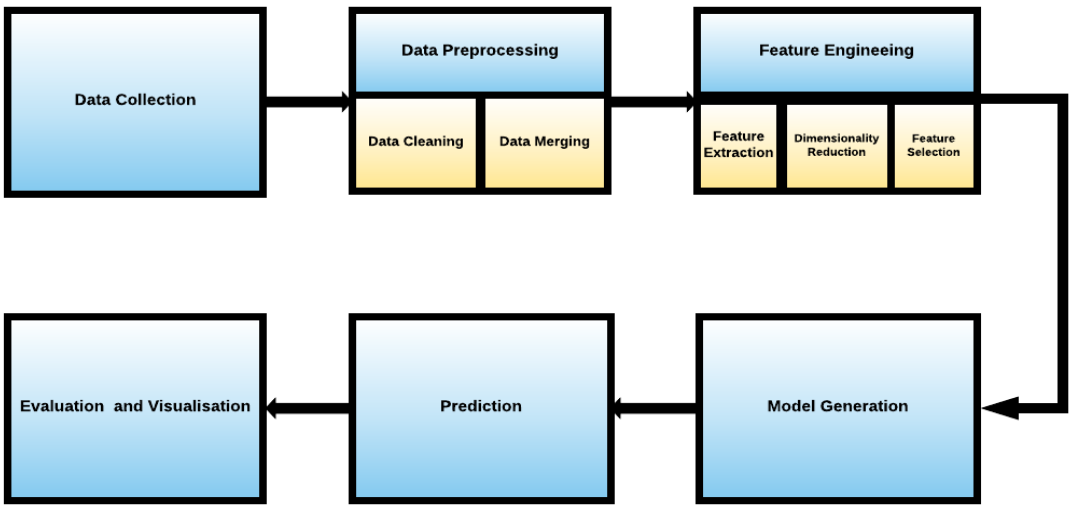
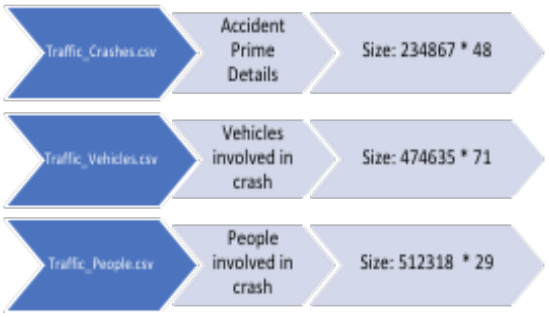


Figure 1: Project Workflow

CHAPTER - 3

About Dataset

Dataset used in this project is provided by Chicago Data Portal. It contains information on road accidents that happened in Chicago in the years of 2015 – 2018. Data is provided as three CSV files.



'Primary Contributory Cause of the Accident' , which is the target variable comes with 38 classes.

Link to the source dataset:
https://data.cityofchicago.org/Transportation/Traffic-Crashes-Crashes/85ca-t3if
https://data.cityofchicago.org/Transportation/Traffic-Crashes-Vehicles/68nd-jvt3
https://data.cityofchicago.org/Transportation/Traffic-Crashes-People/u6pd-qa9d

Data Preprocessing:

Data data obtained from the three CSV files were loaded as dataframes using pandas, and preliminary analysis were made. Processing of data was performed in three steps as described below:

Preprocessing Step 1 - Cleaning:

1) Irrelevant features like damage caused by accident were removed
2) Imputing missing values: Missing values were imputed
3) Observations with too many missing values were dropped.
4) For categorical features, based on other feature values, same values for features with same kind or extracting values were imputed as applicable.
5) Data with incorrect values like lane counts in range of 100,000 were removed.
6) Few new features are constructed based on other features
   Example: date included as one feature is split into day of month, month and year features

Preprocessing Step 2:

1) Three cleaned data frames obtained from the above step was merged into a single dataframe.
2) Details of the contributing factors due vehicle defect, driver behaviors were present in across multiple rows. All the information associated with one crash ID was required to be

present as a single sample. Hence, datasets were merged based on the unique IDs identified in three files.

3) After merging, it was identified that some of the features are not applicable to all samples and hence are seen as null. These null values were further handled in the same approach as described in step 1.

Preprocessing Step 3:
One hot encoding: With this technique, categorical features are binarized and included as separate columns.

Preprocessing Step 4:
The data was split into train and test data in a ratio of 80:20, and is used to train and different classifiers. Since, data is imbalanced, stratified splitting was done.

Preprocessing Step 5:
Data from preprocessing step 4 is scaled to bring values on a uniform scale.

Preprocessing Step 6:

1) After experimenting with various dimensionality reduction/ feature selection techniques, it was narrowed down to three techniques - Principal Component Analysis, Variance Threshold, Truncated Singular Value Decomposition. These feature selection techniques were used on the normalized/scaled data.
2) Since variance threshold was giving better predictions, cross-validation was done using variance threshold.

Methodology

Different the following approaches were followed to build training models.

Approach 1: Without Resampling

1) Data was split into train and test parts in a ratio of 80: 20.
2) For each of the three dimensionality reduction techniques mentioned in "*preprocessing step 5"*, all the algorithms mentioned in "*Part 2 - Algorithm Selection" of this report*.
3) Pipeline mechanism was used to build a queue for the required processing techniques.
4) Evaluation parameters accuracy, F1 score, precision and recall were obtained for each model.

Approach 2: With Resampling

1) To obtain better F1-score, train data was resampled based on the distribution. Oversampling of minority classes and undersampling of majority classes were used to resample data. Different classification models were used to train the resampled data.
2) Evaluation parameters accuracy, F1 score, precision and recall were calculated.

K-fold validation:
10-fold cross validation was performed on the entire dataset using the best performing six algorithms, based on f1-score. Variance threshold was used for feature selection. Multi-Layer Perceptron, Bagging with Decision Trees, Linear Discriminant Analysis, Extra Trees Classifier, Logistic Regression, Random Forest CBassifier are the algorithms used; mean value of the scores obtained were noted.

Results Analysis

Performance of each model was evaluated from various evaluation parameters. Following are the visual representations of the obtained results.

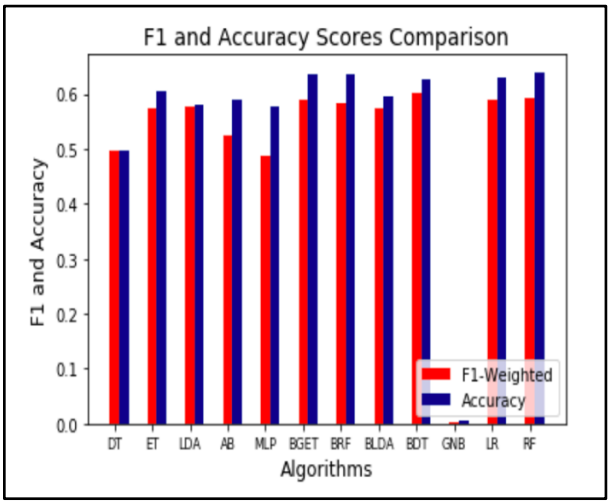


Figure 2: F1 and Accuracy comparison between various classifiers.
Data is *standardized* and features are selected. using *Variance Threshold*

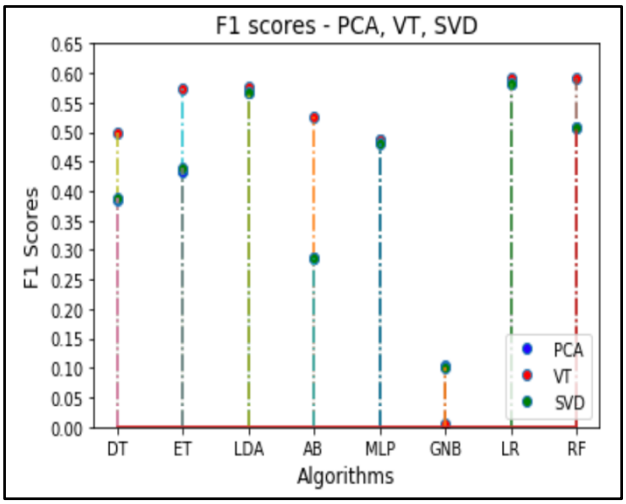


Figure 3: F1 scores comparison between data applied on PCA, VT and SVD dimensionality reduction techniques

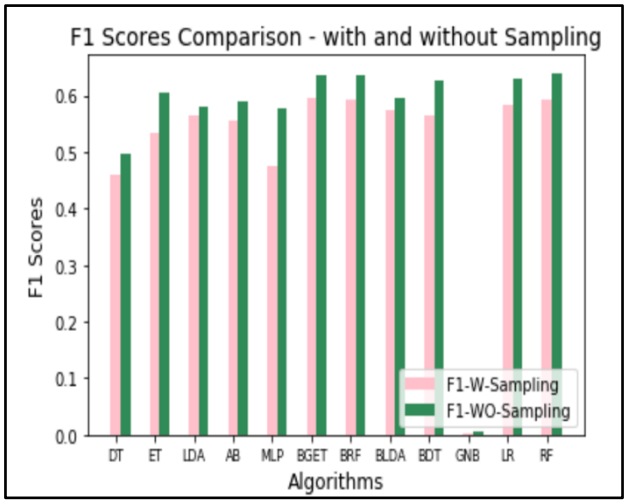


Figure 4: F1 scores comparison between data with sampling and data without sampling

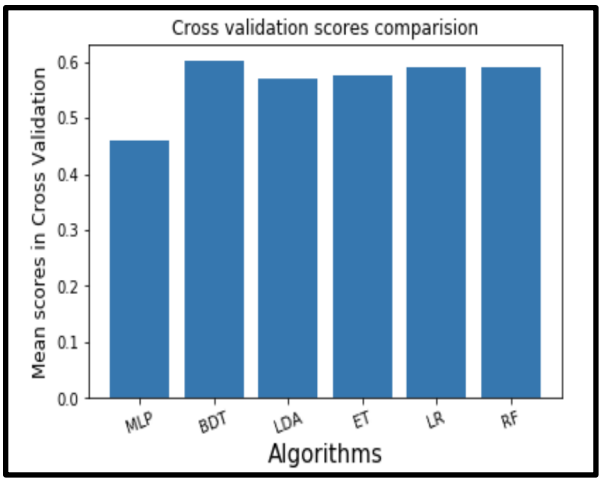


Figure 5: Cross validation mean scores comparison between classifiers

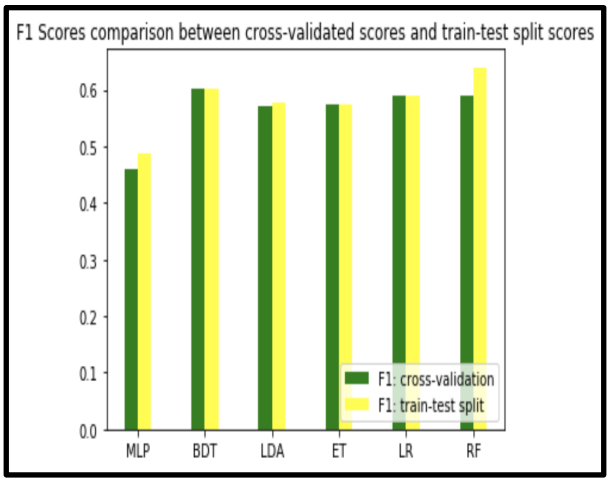


Figure 6: F1 comparison between cross validation and train test split

From figure (2), it is seen that bagging classifiers handles imbalance classes more efficiently when compared to other classifiers. Bagging with Random Forest, bagging with Linear Discriminant Analysis, bagging with Extra Trees Classifier and bagging with Decision Trees provide better F1 scores amongst all.

From figure (3) it is seen that out of the three dimensionality reduction techniques, Variance Threshold provided better results for most of the classifiers.

Data is highly imbalanced and consists of majority of categorical features, because of which even with bagging the F1 score achieved in not very high.

Resampling the data did not prove to be a good strategy to improve F1 score. This is seen in figure (4).

Figures (5) and (6) shows results obtained for cross validation. Cross validation scores closely resemble train-test split results for most of the classifiers.

Numerical results:

Algorithm Name	Normalization /Standardization	Micro_F1_score	Weighted_F1_Score	Precision	Recall
Random Forest	Standardization	0.6396828441	0.5913565821	0.6195473528	0.6396828441
Naïve Bayes	Standardization	0.0054564443	0.0038950982	0.2084925981	0.0054564443
Logistic Regression	Standardization	0.6293454398	0.5898537709	0.5899418346	0.6293454398
AdaBoostClassifier	Standardization	0.5891254769	0.5243464839	0.4857090062	0.5891254769
MLP	Standardization	0.5771042479	0.4866935751	0.4230716256	0.5771042479
Decision Tree	Standardization	0.498007119	0.4975877205	0.4974183013	0.498007119
Extra Tree	Standardization	0.6062621225	0.574788559	0.565983915	0.6062621225
LDA	Standardization	0.5794061854	0.5761158786	0.5979898905	0.5794061854
Bagging with ET	Standardization	0.6368267366	0.588739175	0.6149740094	0.6368267366
Bagging with RF	Standardization	0.6347805699	0.5826099157	0.6196219322	0.6347805699
Bagging with LDA	Standardization	0.5966494021	0.5737261052	0.6060210255	0.5966494021
Bagging with DT	Standardization	0.6267877315	0.6008168379	0.5926255865	0.6267877315

Table 1: *F1_micro, F1_weighted, Precision and Recall* scores for various classifiers. Data is *standardized* and dimensionality is reduced using *Variance Threshold*

CHAPTER - 4

Difficulties faced

1) The training data was present in three different files, with contributory factors in all three, and also in multiple rows. Some of the column names were acronyms. The data source website, google-search did not have any description or information about these acronyms. It was challenging to find their impact on contributory factors. [Two methods, one using Python's merge and the second using traditional loops were implemented to be sure on the combination]
2) Domain knowledge of vehicles, road conditions, weather conditions was required, which made understanding their effects in causing accidents difficult.
3) Many records with missing values did not have direct relation to the identified contributory factors. This decreased the prediction quality.
4) Some of the algorithms like KNN and SVM were running for more than 20 hours, even with different parameters. We tried running on hpc, but the issue persisted. So, we couldn't use these prediction algorithms.

Things that worked

1) To improve F1 score, re-sampling technique was tried. However, it did not increase the prediction quality. Consequently, more trials were made with other bagging classifiers and boosting which increased the prediction quality and were fast.
2) Techniques like pickling were used to store evaluation metrics results. This helped to draw graphs.

Things that did not work well

1) Given the large size of the data, training using boosting and bagging algorithms consumed long time. Consequently, tuning of hyper parameters was limited to fewer trials.
2) Since, the data is highly imbalanced, different resampling techniques like oversampling, undersampling, smote were used. But resampling didn't increase the F1-score.
3) For a few algorithms, like KNN and SVM, it took more than 20 hours, with laptops shutting down. Running these algorithms on HPC did not reduce time taken.
4) When the algorithms mentioned above was run on HPC, connection broke intermittently. Hence , it was required to re-run the models.

Conclusion

The traffic accidents data was analyzed, and different data mining techniques were applied to predict the contributory causes of accidents. Based on these factors, it is possible to predict crashes. This information can also be used by the government to implement new policies, regulations, improve the infrastructure. If missing values were handled by adding more information from the crash-site, the prediction algorithm can be used to improve prediction quality. It was observed from the results that Bagging classifier with Decision Tree as base estimator had the highest F1-score of 0.60.

Individual Contributions

Name	Tasks
Chaithra Lakshmi Sathyanarayana	● Merging of the files in the initial step (approach 1) ● Preliminary analysis of dataset and understanding features ● Strategies to handle bad data ● Identifying better techniques for dimensionality reduction - Variance threshold ● Researching on cross validation ● Preparation of demo data ● Experimenting on various sampling procedures - smote, random over sampling ● Classification algorithms: Linear Discriminant Analysis Decision Tree Classifier Bagging Classifiers Extra Trees Classifier SVM ● Preparation of report ● Preparation of presentation slides
Sayali Pisal	● Preliminary analysis of dataset and understanding features ● Coming up with unique ways to impute missing values in dataset ● Strategies to handle bad data ● Identifying better techniques for dimensionality reduction - PCA ● Implementing cross validation and gridsearch CV ● Setting up HPC and running codes in that environment ● Creation of types of plots for results ● Classification algorithms: Random Forest Classifier Gaussian Naive Bayes Logistic Regression KNN ● Preparation of report ● Preparation of presentation slides
Thaijasa Badrinath Vijendranath	● Preliminary analysis of dataset and understanding features ● Coming up with unique ways to impute missing values in dataset ● Applying one-hot encoding ● Identifying better techniques for dimensionality reduction - TSVD ● Implementing cross validation and gridsearch CV ● Experimenting on various sampling procedures -

	oversampling, under sampling • Creation of types of plots for results • Merging of the files in the initial step (approach 2) • Classification algorithms: Adaboost Classifier Bagging Classifiers Multilayer Perceptron KNN • Preparation of report • Preparation of presentation slides

References

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.multiclass
https://matplotlib.org/gallery/lines_bars_and_markers/barchart.html#sphx-glr-gallery-lines-bars-and-markers-barchart-py
https://stackoverflow.com/questions/41538681/plot-two-lists-with-different-color-with-stem
https://en.wikipedia.org/wiki/List_of_countries_by_traffic-related_death_rate
https://irjet.net/archives/V4/i4/IRJET-V4I4306.pdf
https://www.sciencedirect.com/science/article/abs/pii/S0001457517303913
https://www.sciencedirect.com/science/article/abs/pii/S0001457517303913
https://en.wikipedia.org/wiki/Motor_vehicle_fatality_rate_in_U.S._by_year

Appendix

	Algorithm Name	Normalization /Standardization	Feature scaling /Dimensionality Reduction	Micro_F1_score	Weighted_F1_Score	Precision	Recall
1	Random Forest	Standardization	PCA(n_components=500)	0.5677686126	0.5095599855	0.5517462254	0.5677686126
2	Random Forest	Standardization	Variance Threshold	0.6396828441	0.5913565821	0.6195473528	0.6396828441
3	Random Forest	Standardization	Truncated SVD(n_components=500)	0.5661913592	0.5083101415	0.5529675523	0.5661913592
4	Naïve Bayes	Standardization	PCA(n_components=500)	0.0712321759	0.1050707357	0.3217250962	0.0712321759
5	Naïve Bayes	Standardization	Variance Threshold	0.0054564443	0.0038950982	0.2084925981	0.0054564443
6	Naïve Bayes	Standardization	Truncated SVD(n_components=500)	0.0694417801	0.1022414342	0.3192906519	0.0694417801
7	Logistic Regression	Standardization	PCA(n_components=500)	0.6232495683	0.5808737949	0.5805246278	0.6232495683
8	Logistic Regression	Standardization	Variance Threshold	0.6293454398	0.5898537709	0.5899418346	0.6293454398
9	Logistic Regression	Standardization	Truncated SVD(n_components=500)	0.6235905961	0.5817147236	0.5815207708	0.6235905961
10	AdaBoostClassifier	Standardization	PCA(n_components=500)	0.3713153015	0.2872352851	0.2963381488	0.3713153015
11	AdaBoostClassifier	Standardization	Variance Threshold	0.5891254769	0.5243464839	0.4857090062	0.5891254769
12	AdaBoostClassifier	Standardization	Truncated SVD(n_components=500)	0.3903062856	0.2874573405	0.3168707979	0.3903062856
13	Bagging with KNN	Standardization	PCA(n_components=500)	0.5130762836	0.4668767726	0.4793553668	0.5130762836
16	MLP	Standardization	PCA(n_components=500)	0.5741415692	0.4866327189	0.42636023	0.5741415692
17	MLP	Standardization	Variance Threshold	0.5771042479	0.4866935751	0.4230716256	0.5771042479
18	MLP	Standardization	Truncated SVD(n_components=500)	0.5691114095	0.4797661629	0.4280093495	0.5691114095
19	Decision Tree	Standardization	PCA(n_components=500)	0.3813116781	0.3838214837	0.3866166453	0.3813116781
20	Decision Tree	Standardization	Variance Threshold	0.498007119	0.4975877205	0.4974183013	0.498007119
21	Decision Tree	Standardization	Truncated SVD(n_components=500)	0.3874714922	0.3894516152	0.3916086736	0.3874714922
22	Extra Tree	Standardization	PCA(n_components=500)	0.4753074579	0.4383104118	0.4331224113	0.4753074579
23	Extra Tree	Standardization	Variance Threshold	0.6062621225	0.574788559	0.565983915	0.6062621225
24	Extra Tree	Standardization	Truncated SVD(n_components=500)	0.4735596905	0.4365586507	0.4330452682	0.4735596905
25	LDA	Standardization	PCA(n_components=500)	0.5794061854	0.5670648792	0.5809650613	0.5746317966
26	LDA	Standardization	Variance Threshold	0.5794061854	0.5761158786	0.5979898905	0.5794061854
27	LDA	Standardization	Truncated SVD(n_components=500)	0.5747170535	0.5668843744	0.5806689639	0.5747170535
28	Bagging with ET	Standardization	Variance Threshold	0.6368267366	0.588739175	0.6149740094	0.6368267366
29	Bagging with RF	Standardization	Variance Threshold	0.6347805699	0.5826099157	0.6196219322	0.6347805699
30	Bagging with LDA	Standardization	Variance Threshold	0.5966494021	0.5737261052	0.6060210255	0.5966494021
31	Bagging with DT	Standardization	Variance Threshold	0.6267877315	0.6008168379	0.5926255865	0.6267877315
32	Random Forest	Normalization	PCA	0.6383	0.5489		
33	Naïve Bayes	Normalization	PCA	0.1149	0.1304		
34	Logistic Regression	Normalization	PCA	0.3612	0.3612		
35	AdaBoostClassifier	Normalization	-	0.57	0.57		
36	Bagging with KNN	Normalization	-	0.4390606229	0.4390606229		
37	Bagging with KNN	Normalization	PCA	0.4607236511	0.4607236511		
38	MLP	Normalization	PCA	0.3637115213	0.3637115213		

Table 1.1: *F1_micro, F1_weighted, Precision and Recall* scores for various classifiers.