Glass Type Identification Based on Minerals Present, using classification techniques

Sree Leela Nadipalli
Masters in Computer Science
sree.leela92@gmail.com
2515910469
San Diego State University
5500 Campanile Dr, San Diego, CA 92182, United States

***Abstract* — Depending on the requirement, there are different types of glasses used for various purposes. The glass type depends on the minerals present and the proportion in which these minerals are present. Other important feature of glass is Refractive Index. These features contribute to the physical and chemical properties of the glass, and makes it suitable for a specific purpose. So, based on refractive index and mineral proportion we can identify the type of class. In this paper, we used different machine learning classification techniques to classify the type of glass. We also compared different classification techniques to find the best fit.**

***Index Terms* —**
DTC - Decision Tree Classifier
GNB - GaussianNB
KNB - KNeighbors Classifier
LR - Logistic Regression
LDA - Linear Discriminant Analysis
MLP - Multilayer Perceptron Classifier
RF - Random Forest Classifier
SVM - Support Vector Machine

I. Introduction

For implementing glass type identification using machine learning algorithms we follow a series of steps:

1) We take dataset and analyze it. We see how well the parameters are correlated.
2) Next we predict the type of glass using eight machine learning techniques. Three algorithms are discussed in detail. Remaining algorithms comparison graph is given in conclusion.
3) We then compare which is best fit and which has greater accuracy.

II. Dataset (Features and Classes)

The dataset is taken from kaggale.com and it has nine features. They are RI – Refractive Index and other eight minerals compositions, Na – Sodium, Mg – Magnesium, Al – Aluminum, Si – Silicon, K – Potassium, Ca – Calcium, Ba – Barium, and Fe – Iron.

A. *Vectorization of Features*

Generally, machine learning methods require the features to be numeric. Therefore, we need to convert them appropriately. But, in the dataset used here, all the features are numeric and no specific dealing with dataset is required.

B. *Analyzing Dataset*

Analyzing the dataset helps to understand how well the data is correlated and how accurate the prediction would be. If the data is related nicely, then the accuracy will be high else we may expect low accuracy.

Below histograms and boxplots of all the features are shown. However, to understand and visualize the correlation between features, a correlation matrix would help us a lot. So, a correlation matrix is also given below.

The graphs show the amount of each mineral present and its frequency in the dataset.

X axis – Composition of mineral /Quantity of mineral.
Y axis – number of rows containing that composition in dataset/frequency of the mineral in the dataset.

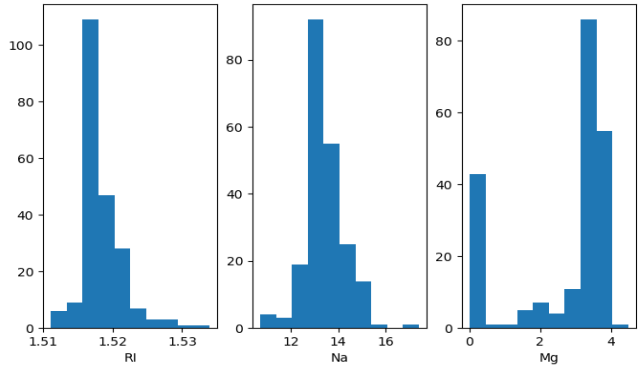

Figure 1: Histogram of data showing RI, Na, and Mg in dataset

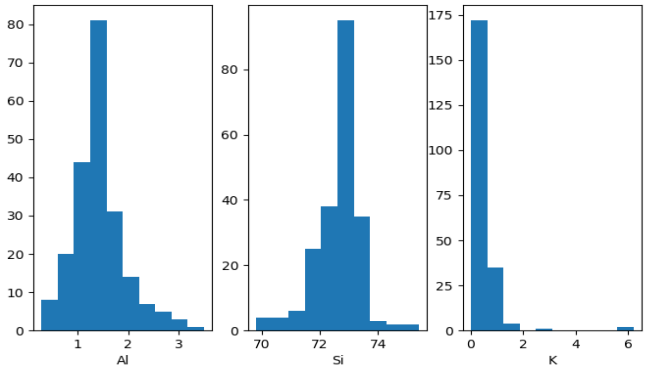

Figure 2: Histogram of data showing Al, Si, and K in dataset

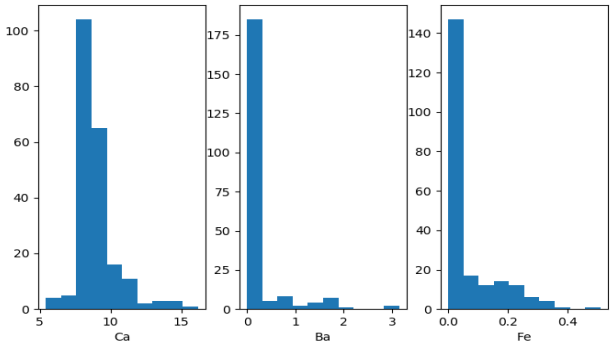

Figure 3: Histogram of data showing Ca, Ba, and Fe in dataset

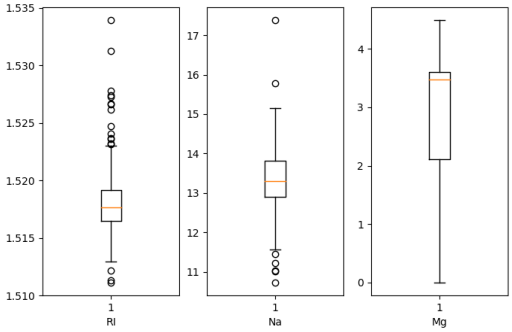

Figure 4: Boxplot of data showing RI, Na, and Mg in dataset

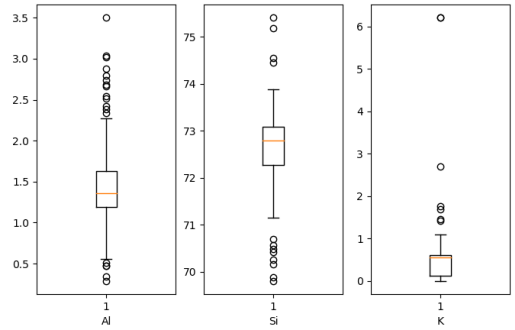

Figure 5: Boxplot of data showing Al, Si, and K in dataset

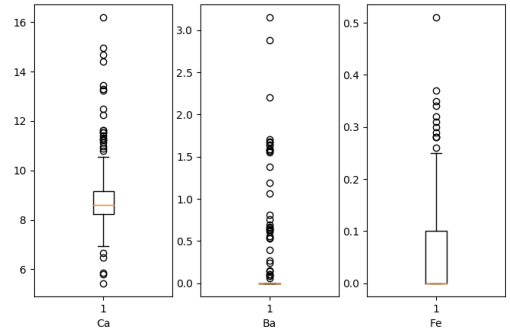

Figure 6: Boxplot of data showing Ca, Ba, and Fe in dataset

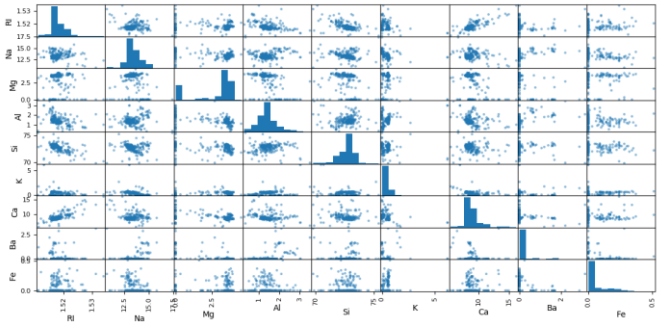

Figure 7: Scatter plot of data in dataset

From the histogram, boxplot and scatter plot we see the range of data. Now, let us see the correlation matrix to understand how the data is dependent on each other.

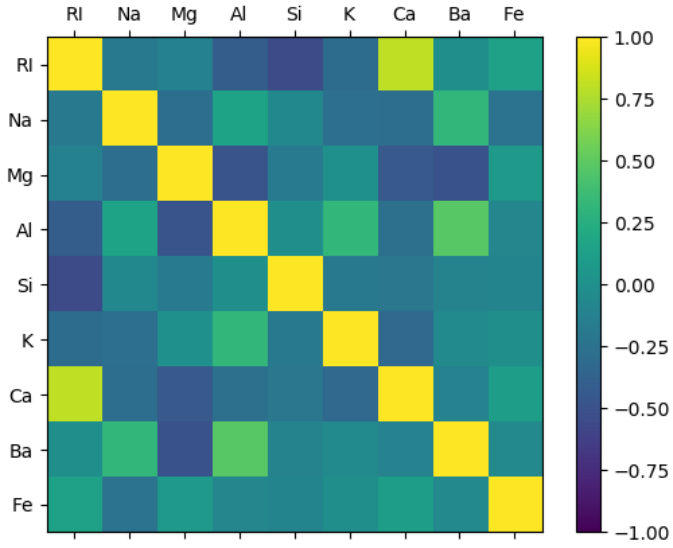

Figure 8: Correlation matrix of data in dataset

From correlation matrix, it is clear that the attributes in the dataset are not so well correlated and accuracy may be low.

C. Classes

The classes range from 1 to 7. Each number represents different type of glass. The below table shows class and glass type.

TABLE I
CLASS AND ITS GLASS TYPE

Class	Glass Type
1	Building windows float processed
2	Building windows non float processed
3	Vehicle windows float processed
4	Vehicle windows non float processed (not present in this database)
5	containers
6	tableware
7	headlamps

D. *Cross Validation*

To make sure that a model is not evaluated with the same data on which it was trained, cross validation is implemented by using a method called K-Fold Cross Validation here.

III. Algorithms and Analysis

A. *Logistic Regression*

Logistic regression is a generalized linear model for classification with a probabilistic interpretation that can be used to predict discrete values.

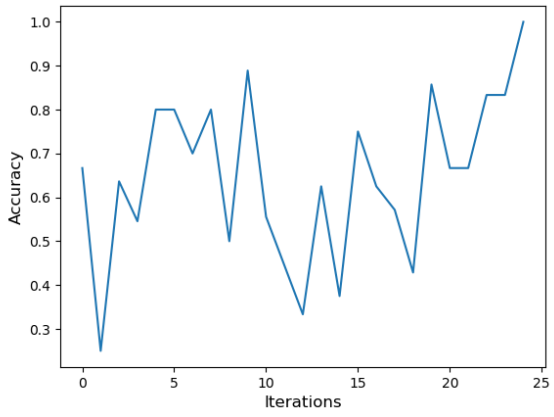

Figure 9
Accuracy and Iterations for Logistic Regression

From figure 9 shown above, it is clear that as the iterations increases, the accuracy increases. The significant parameters from logistic regression are given below:

- C : float, default: 1.0
  Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization.
- solver : {'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'}, default: 'liblinear' Algorithm to use in the optimization problem.
- max_iter : int, default: 100
  Useful only for the newton-cg, sag and lbfgs solvers. Maximum number of iterations taken for the solvers to converge.
- random_state : int, RandomState instance or None, optional, default: None
- fit_intercept : bool, default: True
  Specifies if a constant (a.k.a. bias or intercept) should be added to the decision function.
- penalty : str, 'l1' or 'l2', default: 'l2'
  Used to specify the norm used in the penalization. The 'newton-cg', 'sag' and 'lbfgs' solvers support only l2 penalties.
- tol : float, default: 1e-4
  Tolerance for stopping criteria.
- intercept_scaling : float, default 1.
  Useful only when the solver 'liblinear' is used and self.fit_intercept is set to True.

TABLE II
PARAMETERS FOR LOGISTIC REGRESSION

PARAMETERS	ACCURACY
C=1, solver='liblinear', random_state=1	0.619808080808
C=10, solver='liblinear', penalty='l2'	0.641144300144
C=0.0125, solver='liblinear', penalty='l2', max_iter=5	0.552093795094
C=0.125, solver='lbfgs', fit_intercept=0	0.601952380952
C=10, solver='lbfgs', multi_class ='ovr'	0.646113997114
C=10, solver='liblinear'	0.641144300144
intercept_scaling=0.13, fit_intercept=1, penalty='l2'	0.619808080808
C=10, solver='liblinear', max_iter=15, class_weight='balanced'	0.617513708514

From the above table II, it is clear that accuracy increases as C increases. Accuracy also depends greatly on iterations. As the number of iterations increase, accuracy increase. If less iterations are given then accuracy decreases. Sufficient iterations are needed to produce efficient results with logistic regression.

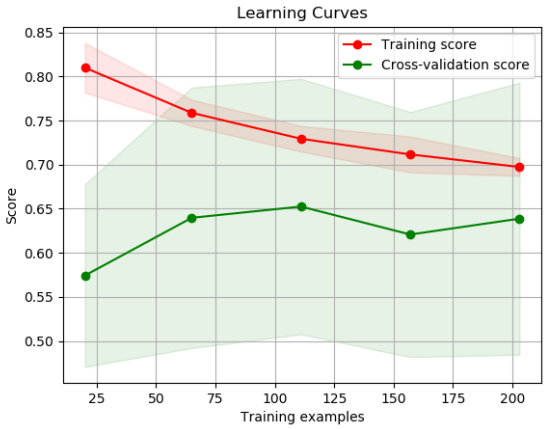

Figure 10
Learning Curves for Logistic Regression

From figure 10 show above, it is observed that the learning curves, training score and cross validation score converge very well for logistic regression.

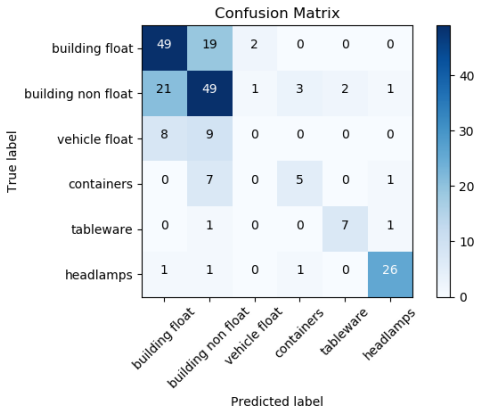

Figure 11
Confusion Matrix for Logistic Regression

Overall classification report is given in table III which shows Recall, Precision, F1-score and Support for six different types of glass.

TABLE III
CLASSIFICATION REPORT FOR LOGISTIC REGRESSION

	Precision	Recall	F1-score	support
Building Float	0.62	0.70	0.66	70
Building non float	0.57	0.64	0.60	77
Vehicle float	0.00	0.00	0.00	17
Containers	0.56	0.38	0.45	13
Tableware	0.78	0.78	0.78	9
Headlamps	0.90	0.90	0.90	29
Avg/total	0.59	0.63	0.61	215

B. Random Forest Classifier

A random forest is a Meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

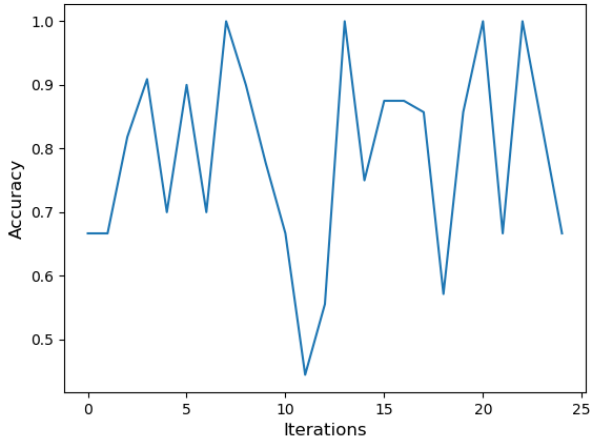

Figure 12
Accuracy and Iterations for Random Forest Classifier

From figure 12, it is observed that as iterations increase, accuracy is improved till certain point. After that the accuracy doesn't show significant increase or the accuracy decreases. Significant parameters in Random forest classifier are:

- *n_estimators*: This is the number of trees you want to build before taking the maximum voting or predictions. Higher number of trees give better performance but makes code slower.
- *min_samples_leaf*: If you have built a decision tree before, you can appreciate the importance of minimum sample leaf size. Leaf is the end node of decision tree.

The above parameters are studied and experimented and the results are shown in table IV below. n_estimators didn't contribute to accuracy much. However, as min_leaf_samples increased the accuracy decreased.

TABLE IV
PARAMETERS FOR RANDOM FOREST CLASSIFIER

N_ESTIMATORS	MIN_SAMPLES_LEAF	ACCURACY
10	1	0.759316017316
20	1	0.771265512265
35	1	0.789216450216
20	15	0.662656565657
35	15	0.653861471861

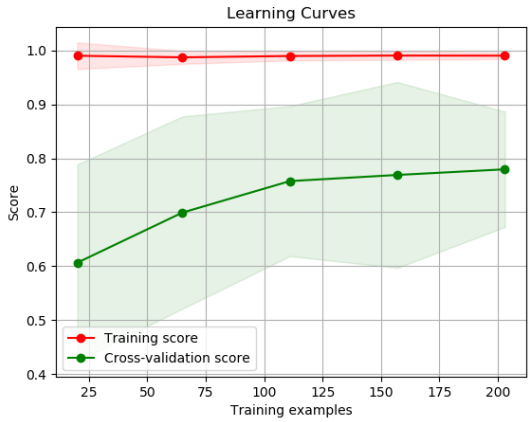

Figure 13
Learning Curves of Random forest Classifier

From figure 13 show above, it is observed that the learning curves, training score and cross validation score converge slowly. Even though the learning curves are converging slowly, it is observed that the accuracy of this model is effective.

Since the learning curves converge slowly, if this model is given with more samples, then there are very high chances that this model accuracy increases.

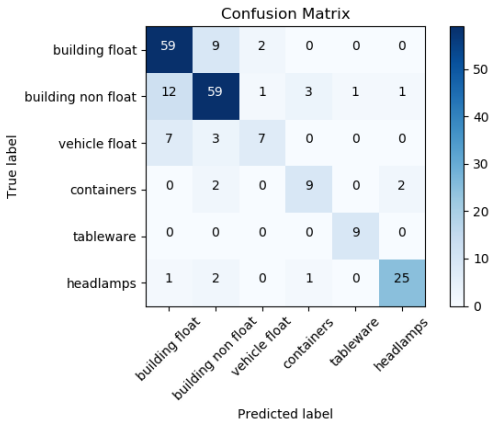

Figure 14
Confusion Matrix of Random Forest Classifier

Figure 14, shows the confusion matrix for random forest classifier. It clearly shows that this model did have good accuracy when compared to others. Overall classification report is given in table V which shows Recall, Precision, F1-score and Support for six different types of glass.

TABLE V
CLASSIFICATION REPORT FOR RANDOM FOREST CLASSIFIER

	Precision	Recall	F1-score	support
Building Float	0.71	0.87	0.78	70
Building non float	0.78	0.75	0.77	77
Vehicle float	0.33	0.18	0.23	17
Containers	0.83	0.77	0.80	13
Tableware	1.00	0.78	0.88	9
Headlamps	0.89	0.83	0.86	29
Avg/total	0.75	0.76	0.75	215

C. Support Vector Machine

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers' detection. In this project we are using Scikit-learn package for implementing SVM algorithm. The support vector machines in Scikit-learn support both dense (numpy.ndarray and convertible to that by numpy.asarray) and sparse (any scipy.sparse) sample vectors as input. [5].

We are using Linear SVC [6] i.e. Support Vector Clustering mechanism which is similar to SVC with parameter kernel='linear', but implemented in terms of "liblinear" rather than libsvm, so it has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples.

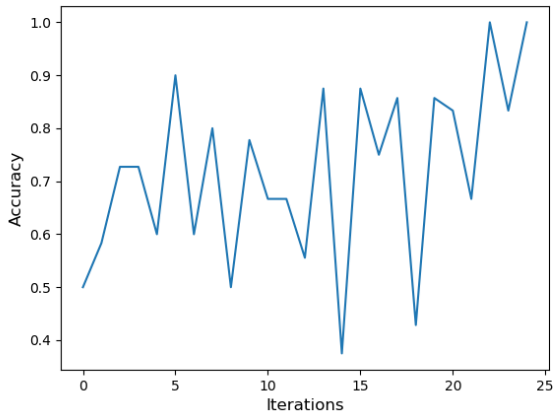

Figure 15
Accuracy and Iterations for SVM

From the above graph it is clear that the accuracy increases as the iterations increase. This is also shown in the table VI. The significant parameter for SVC are:

- C: float, optional (default=1.0) Penalty parameter C of the error term.
- kernel: string, optional (default='rbf'). Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable.
- max_iter: int, optional (default=-1) Hard limit on iterations within solver, or -1 for no limit.
- random_state : int, RandomState instance or None, optional (default=None)
- Coef0: array, shape = [n_class-1, n_features] Weights assigned to the features (coefficients in the primal problem). This is only available in the case of a linear kernel. coef_ is a readonly property derived from dual_coef_ and support_vectors_.

The above parameters are studied and experimented. The results are shown in the table VI. If kernel is rbf, it has highest accuracy. Next comes poly kernel and then comes linear kernel. Sigmoid has least accuracy. And in all the kernels, as the iterations increase the accuracy increases.

In linear kernel, C does not have major effect on accuracy but it does have minor effect. In poly, Coef0 does increase the accuracy.

TABLE VI
PARAMETERS FOR SUPPORT VECTOR MACHINE

PARAMETERS	ACCURACY
kernel='linear'	0.64575036075
kernel='linear',C=1	0.64575036075
kernel='linear',C=100	0.682797979798
kernel='linear',C=1000	0.670464646465
kernel='linear',C=0.0001	0.373246753247
kernel='linear',C=0.1	0.626538239538
kernel='linear',C=1, max_iter=10	0.511653679654
kernel='linear',C=1, max_iter=100	0.60232178932
kernel='linear',C=1, max_iter=1000	0.64575036075
kernel='linear',C=100,max_iter=10000	0.665909090909
kernel='linear',C=100,max_iter=100000	0.682797979798
kernel='linear',C=100,max_iter=1000000	0.682797979798
kernel='poly'	0.709454545455
kernel='poly',C=1	0.709454545455
kernel='poly',C=100	0.695404040404
kernel='poly',C=1000	0.697608946609
kernel='poly',C=0.1	0.695815295815
kernel='poly',C=0.0001	0.695815295815
kernel='poly', max_iter=10	0.485727272727
kernel='poly', max_iter=100	0.573132756133
kernel='poly', max_iter=1000	0.709454545455
kernel='poly', max_iter=10000	0.709454545455
kernel='poly', coef0=1e-4	0.709454545455
kernel='poly', coef0=1	**0.719454545455**
kernel='poly', coef0=3	0.698484848485
kernel='poly', coef0=5	0.711865800866
kernel='poly', coef0=0.1	0.709454545455
kernel='poly', coef0=0.0001	0.709454545455
kernel='rbf'	**0.718229437229**
kernel='rbf', C=100	0.687705627706
kernel='rbf', C=0.01	0.373246753247
kernel='rbf', max_iter=10	0.503367965368
kernel='rbf', max_iter=10000	0.718229437229
kernel='sigmoid'	0.373246753247
kernel='sigmoid',C=100	0.345357864358

Rbf kernel gives effective results. However, poly kernel with good coef0 gives better results than rbf.

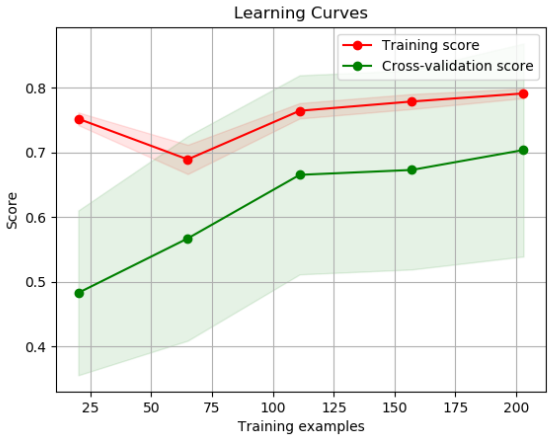

Figure 16
Learning Curves for SVM

The learning curves, training score and cross validation score eventually converge. The algorithm performs even better, if more samples are given.

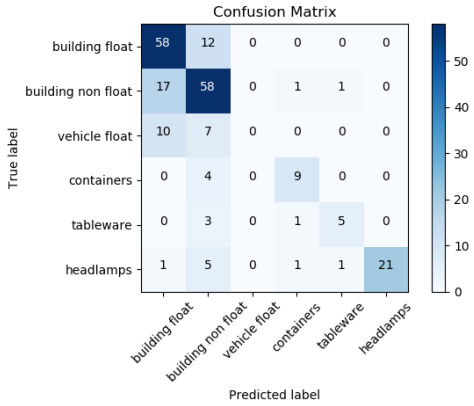

Figure 17
Confusion Matrix for SVM

Overall classification report is given in table VII which shows Recall, Precision, F1-score and Support for six different types of glass.

TABLE VII
CLASSIFICATION REPORT FOR SUPPORT VECTOR MACHINE

	Precision	Recall	F1-score	support
Building Float	0.67	0.83	0.74	70
Building non float	0.65	0.75	0.70	77
Vehicle float	0.00	0.00	0.00	17
Containers	0.75	0.69	0.72	13
Tableware	0.71	0.56	0.63	9
Headlamps	1.00	0.72	0.84	29
Avg/total	0.66	0.70	0.68	215

IV. Major Challenges and Solutions

The dataset was small and the accuracy was low initially. So it was time consuming and difficult to dig down various other algorithms to improve accuracy and learning curve convergence to get effective results. Not only different algorithms, but also different ways each algorithm can be implemented was deeply studied to get to improve learning rate and get effective results. Finally, after regressive study Random forest classifier and SVM seemed to be best fit and it was further experimented as discussed above.

There were some issues faced while installing scikit learn on the Windows operating system.

The dataset was not only small but also has very low correlation rate. Since most of the features are not correlated it gave low accuracy from most of the models. It took some time to analyze data and different types of graphs were generated from dataset to analyze the data as discussed earlier. Thus, by overcoming these challenges, good results were generated.

V. Conclusion and Analysis

The analysis and conclusion is done based on the accuracy and other information obtained from the graphs of each model for the given classification problem.

From all the eight machine learning algorithms implemented in this paper, ***Random Forest Classifier*** gives effective results and best accuracy amongst all. Its accuracy is nearly 80%. Next comes ***Support Vector Machine*** which is nearly equal to random forest classifier and then comes ***MLPClassifier*** and ***Decision Tree Classifier*** which gives almost the same accuracy but little less accurate than Random forest classifier and support vector machine. ***Logistic Regression***, ***Linear Discriminant Analysis*** and ***KNeighbor Classifier*** gives almost the same accuracy but less accurate when compared to the models discussed above. ***GaussianNB*** gives the least accuracy.

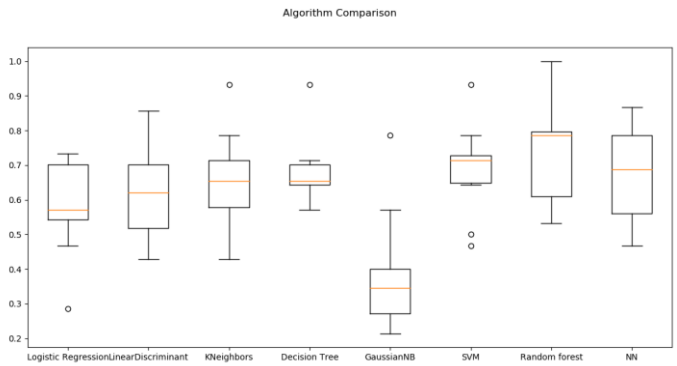

Figure 18
Boxplot Comparison of Eight Machine Learning Algorithms implemented in this paper

The above boxplot shows the accuracy plot of eight algorithms. It clearly shows random forest model gives best results when compared to other algorithms. In SVM, as discussed earlier, the learning curves, that the Training score and the Cross-validation scores are ultimately converging. Thus, the training score is still around the ideal point and the validation score could further be improved with more samples.

VI. Future Work

Future work includes evaluation of these machine learning algorithms on a larger dataset with more classes, to obtain an improved accuracy. We can also perform content-based recommendation of research papers by analyzing the above techniques upon generalization over large datasets.

References

[1] Rajaraman, A.; Ullman, J. D. (2011). "Data Mining". Mining of Massive Datasets (PDF). pp. 1–17.

[2] "4.2. Feature extraction — scikit-learn 0.18.1 documentation", Scikit-learn.org. [Online]. Available: http://scikit-learn.org/stable/modules/feature_extraction.html. [Accessed: 04- May- 2017].

[3] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning*, 1st ed. New York: Springer, 2016. pp. 242

[4] http://www.csie.ntu.edu.tw/~cjlin/liblinear/

[5] http://scikit-learn.org/stable/modules/svm.html#svc

[6] http://scikitlearn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

[7] http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

[8] https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.stats.linregress.html

[9] http://scipy-cookbook.readthedocs.io/items/LinearRegression.html

[10] https://www.datasciencecentral.com/profiles/blogs/linear-regression-in-python-use-of-numpy-scipy-and-statsmodels

[11] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[12] Jason Brownlee; [November 14, 2016], "Random Forest Model From Scratch", [https://machinelearningmastery.com/implement-random-forest-scratch-python/]

[13] Yhat; [june 5; 2013], "Random Forest in Python", http://blog.yhat.com/posts/random-forests-in-python.html

[14] http://scikitlearn.org/stable/modules/tree.html#classification

[15] http://scikitlearn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

[16] https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/

[17] Scikitlearn.com(http://scikitlearn.org/stable/modules/generated/sklearn.neighbors.NearestCentroid.html)

[18] [22] Documentation of Mathworks.com (https://www.mathworks.com/help/stats/kmeans.html?requestedDomain=www.mathworks.com)

[19] http://scikit-learn.org/stable/modules/naive_bayes.html

[20] http://scikitlearn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html