Project: Classification of Glass for Crime Scene Investigation

Sai Soham Ramagiri

Introduction – Glass Analysis, a real-world Data Mining task at hand:

Broken or shattered glass is a common form of forensic evidence found at crime scenes, but its value is not always fully appreciated. Different types of glass, such as that from windows, headlamps, or tableware, can be identified by their distinctive characteristics and compared to each other. When glass is shattered by a strong force, it can scatter up to nine feet and become embedded in the clothing, footwear, or hair of anyone nearby. This means that glass shards can be used to connect a suspect to the crime scene, the objects found there, and the people present at the time of the crime. Glass fragments are a valuable piece of evidence in many types of crimes, including burglaries, murders, hit-and-runs, and others.

Glass analysis is primarily concerned with classification, discrimination, and individualization of glass evidence. Because different types of evidence samples may be discovered at the same crime scene, it is critical to determine and classify the exact source of each evidence sample.

Formulation as a Classification Problem:

The problem is to forecast the type of glass based on the chemical analysis of its unique properties. This project deals with the **study of various Classification Algorithms** to identify the source of the evidence sample of glass shards that are found at a Crime Scene.

The focus is on identifying the source of glass shards found at crime scenes through the analysis of the unique chemical properties of glass such as refractive index and oxide content and applying classification algorithms on the data. A variety of classification techniques will be tested, and their accuracy compared to determine the most effective method for classifying glass in criminological investigations.

Datasets:

The Glass Identification Dataset is collected from the USA Forensic Science Service Data and downloaded from UCI Machine Learning Repository. This dataset was created by B. German in the Central Research Establishment of USA Forensic Science Service. Vina Spiehler, Ph.D., DABFT had conducted the research for the study of classification of types of glass which was motivated by criminological investigation.

This is a multivariate data frame with 214 records containing instances of the chemical analysis of 7 distinct types of glass – building windows float processed, building windows non float processed, vehicle windows float processed, vehicle windows non float processed (none in this database), containers, tableware, headlamps. There are 9 attributes dealing with various characteristics of glass defined in terms of their refractive index, weight percentage in oxide content for Sodium, Magnesium, Aluminum, Silicon, Potassium, Calcium, Barium, and Iron. This dataset contains no missing values and is essential for this study.

To begin with, we load and read the dataset, and look at its information using data.info(). This information can be analyzed to see if there are any missing values or duplicate values (none in this case). Then, we separate the Class Labels from the dataset by dropping the last column of the dataset ('Type") and saving it as a list of all the Labels. Furthermore, preprocessing of this dataset involves splitting the dataset into training and testing data and using a Standard Scaler to standardize the features by removing mean and scaling to unit variance. The **Preprocessing** is done separately on train and test data since while normalizing, there are chances of data leakage which should be avoided, meaning no information from the test set should be used to preprocess the training set.

Data Mining Algorithms:

Decision Tree Classifier, Random Forest Classifier, k-Nearest Neighbors (kNN) Classifier, Naïve Bayes Classifier, Support Vector Machine Classifier, and Ensemble methods of Bagging and Boosting can be imported from scikit-learn package and applied on the training data to classify the data and the results can be compared for test data to see what kind of classifier works the best on the data.

Results: Performance Comparison:

The accuracy of the various Classification Algorithms has been measured and compared for analyzing which is the state-of-the-art method with a high accuracy for this experiment.

k-Nearest Neighbors:

This method stores the Training Data and classifies a new data point using this Data on the basis of Similarity. When the **k-Nearest Neighbors (kNN) Classification** is employed on the dataset, after Cross-Validation, the best parameters were determined to be **13 neighbors**, with **Manhattan Distance** selected as the best distance metric. This is determined using Randomized search on hyper parameters. **RandomizedSearchCV()** implements a “fit” and a “score” method. The parameters of the estimator used to apply these methods are optimized by cross-validated search over parameter settings. The best score is then determined to be approximately 0.64.

When KNeighborsClassifier() is imported and employed on this training data with the parameters: 7 nearest neighbors, Uniform weights, and Manhattan Distance metric, the final score for this model is determined to be 0.725 or **72.5%** for Training Data and 0.666 or approximately **67%** for Test Data. Hence, we can see that classifying new records is computationally expensive and **choosing the k value prior to Model Building** is also a weakness for this Algorithm.

Decision Tree Classifier:

When RandomizedSearchCV() is used to find the best fit, and the best score and parameters for the dataset, the best parameters were observed to be 10 minimum samples, 10 maximum leaf nodes, maximum depth of 4, and “gini index” criterion. The best score was 0.631.

The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. When the DecisionTreeClassifier() is imported from scikit-learn package, and is fit on the training data set, its accuracy score was **78.75%** for Training Data and **69.3%** for Test Data. This accuracy was **comparatively higher** than that was observed for kNN Classifier.

Random Forest Classifier:

Random forest is a machine learning method that uses an ensemble of decision trees to make predictions. It works by fitting several decision trees on different subsets of the data, and then aggregating their predictions to make a final prediction. Each tree in the ensemble makes a class prediction, and the class with the most votes is the model's final prediction. The use of multiple decision trees helps to reduce overfitting and improve the model's accuracy.

After applying Randomized Search, the best parameters were found to be 150 estimators and Gini index criterion. The model is built by importing RandomForestClassifier() from scikit-learn and applying it on Training Data. The accuracy score for training data was observed as **90%** and that for test data was observed as **70.3737%**. The Model has **high time complexity** and it's computationally intensive. Random forests are also sensitive to the hyperparameters that are used to train the model, such as the number of trees and the maximum depth of each tree. If these hyperparameters are not set properly, the model can perform poorly.

Support Vector Machine:

Then, Support Vector Classifier is employed on this dataset. They are based on the idea of finding a hyperplane in a high-dimensional space that maximally separates different classes. RandomizedSearchCV() is used to find the fit and the best score and parameters for the dataset. The best estimator was found to be for C = 10000.0. Here, C is a Regularization parameter. The regularization strength is said to be inversely proportional to C, and it must strictly be positive. The best parameters were observed to be 'kernel': 'rbf' which is a default value for the kernel type to be used in the algorithm. The best score observed using Randomized Search was approximately 0.60.

The Support Vector machine, SVC() is imported from scikit-learn, and it is fitted onto the training data with C = 10000 or 1E3 and kernel: 'rbf'. The final score for this Classification Model was 0.7 or **70%** for Training Data and 0.574 or **57.4%** for Test Data. The **major problem** for this Algorithm is that of sensitivity to hyperparameters such as kernel type and penalty term. The selection of different kernel functions gives different results, in this case.

Naïve Bayes Classifier:

Gaussian Naive Bayes supports continuous valued features and models each as conforming to a Gaussian distribution. The model is built by importing GaussianNB() from scikit-learn package and it is fitted on the training data. The mean accuracy on the given training data, test data and labels, is determined and the score is 0.49 or **49%** for Training Data and 0.35 or **35%** for Test Data. Its accuracy is very low because it assumes independence between features, is sensitive to irrelevant features, and cannot handle categorical data. Therefore, it can be easily outperformed by other, more sophisticated algorithms.

Ensemble Methods:

Bagging Classifier:

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions by majority voting, to form a final prediction. For Model Building, the Bagging Classifier is imported from scikit-learn package and applied on the training data set, where the base classifiers for the Ensemble Method of Bagging are Decision Tree Classifiers. Here, the accuracy scores for training, and test data were observed as **99.375% and 77.77%** respectively. Hence, we can see that the variance of the base estimator (decision tree) is reduced, by introducing randomization into its construction procedure (bootstrapping with replacement) and then making an ensemble out of it.

Boosting: AdaBoost Classifier:

An AdaBoost Classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are increased such that subsequent classifiers focus more on difficult cases. When the AdaBoostClassifier() is fitted onto the training dataset with base estimator, RandomForestClassifier() for a 100 estimators, the accuracy scores for training and test data were found to be **100%** and that for test data was observed as **74.074%.** This accuracy is also very high since it weights a model's contribution by its performance.

Conclusion – Choosing the most suitable Classification Technique:

After implementation of these Classification Techniques, the accuracy scores for the **Ensemble Methods** of Bagging and Boosting as well as the Random Forest Classifier were observed to be high for both Training and Test datasets compared to those of other classification models. So, rather than making one model and hoping that the model is the best, and the most accurate predictor, it is better to use ensemble methods which take a myriad of models into account and average those models to produce one final model of high accuracy.

In conclusion, we can see that by combining the predictions of multiple models, Ensemble Methods can capture a wider range of patterns in the data, leading to improved performance, and are robust to noise. The disadvantages of using these methods are the increased complexity, computational cost and the difficulty in tuning due to sensitivity to specific models and hyperparameters used. There might also be a lack of transparency as to why a particular prediction was made.

References:

[1.] *Forensic Casework - Glass*. Cellmark Forensics.
https://www.cellmarkforensics.co.uk/services/forensic-casework/glass/ - :~:text=Glass is frequently encountered at,don't alter over time

[2.] *Forensic analysis of glass evidence: Past, Present and Future.* Legal Desire.
https://legaldesire.com/forensic-analysis-of-glass-evidence-past-present-and-future/

[3.] *Forensic glass analysis.* James E. Girard – Criminalistics: forensic science, crime and terrorism 2nd 97-112

[4.] *Glass Classification.* Chetana Thadhani. https://rstudio-pubs-static.s3.amazonaws.com/539767_6a67f031065f4cbaa8d23f637c1db595.html

[5.] *UCI Machine Learning Repository for Dataset:*
https://archive.ics.uci.edu/ml/datasets/Glass+Identification

[6.] *scikit-learn.* Machine Learning in Python using scikit-learn: https://scikit-learn.org/stable/