Computing Challenge 2023

Group member CIDs:

02142956,

02097359,

02042423,

02031302,

02054243

Submitted files:

.ipynb_checkpoints

PNG Interactive Graphs.PNG

Final Submission Code.ipynb

FullData.csv

FullData_os.csv

FullData_Preprocessed.csv

glass_eu.json

glass_uk.csv

glass_us.json

samples.evidence

samples_preprocessed.csv

Code Predictivity Analysis

Pre-processing:

Each file containing the glass data (UK, EU and US) was given in a different format (.csv and .json), so a Formatting class was implemented to enable each file to be converted into a pandas DataFrame, allowing the data to be encoded and scaled. The DataFrames were then concatenated to create one large database with the glass data (206 samples). The next step was to deal with the NaN values within the data, and it was decided to replace them with the mean value of the other values in its column, calculated only from glasses of that same type (i.e. a NaN value in the 'Fe' column for a type 5 glass is replaced by the mean 'Fe' value of only the other type 5 glasses). The mean was chosen to impute the data as using the median or mode could allow outliers to significantly skew the data.

It was also noticed that there were values of 0 located in the 'RI' column of the database. It was decided that a glass having a refractive index of 0 does not make sense in the real world, so we used imputing to replace the 0 values with the mean value of the refractive index for just that type that the glass falls under. (i.e. the same process that is used to replace the NaN values, but in this case for values of 0). This consideration was made given the refractive index of a material is a ratio of the speed of light in a medium to a vacuum.

Encoding

The data is already encoded by the type column, so encoding is not essential. However, the data can be One-hot encoded by the type in order to avoid implementing a hierarchy into the data. Label encoding would introduce an order to the data by sorting it, which is not desired, as the machine learning algorithm associates the higher categorical value as being 'better'. One-hot avoids this issue and was therefore coded into the Pre-processor class as an optional feature.

Scaling

Scaling in machine learning is a very useful tool and very necessary. Scaling brings data points closer together so that the distance between them is reduced. Moreover, the samples are not treated independently since they are all scaled relative to each other. Scaling was crucial for the classifier algorithms such as KNN which is sensitive to the Euclidean distances between data points, and Logistic regression which fails to converge when data is unscaled.[10]

Minmax scaling scales the minimum and maximum values of the data to be 0 and 1 (if given positive values), with every other value between 0 and 1. Minmax scaling preserves the shape of the original distribution, which is desirable in this project, as data points must be kept the same relative to every other data point. Standard scaling scales every value around a mean of 0, and a standard deviation of 1, and is favoured when the data somewhat adheres to a normal curve.[11]

The sample data given is random data, so Minmax encoding works better and was therefore our scaling method of choice. To further justify our choice, Minmax and standard scaling were both implemented and tested, and it was found that Minmax scaling leads to a higher accuracy score after running the classifier.

Oversampling

The use of oversampling was integral to training the algorithm since a particular class did not dominate, therefore the machine learned equally from the decisions. However, we strongly recommend that SMOTE in favour of random oversampling since SMOTE creates new data points based on taking the mean of a few neighbours, so it is able to generate data that is more representative of the whole.

Synthetic Minority Oversampling Technique (SMOTE)

SMOTE generates data for the minority classes by taking the average of k nearest neighbours. This ensures that the amount of data in each class is equal to the majority class. This helped to mitigate the effects of having an imbalanced dataset and led to predictions that were less skewed towards the majority classes.[1] SMOTE was carried out only after scaling the data, since the k nearest neighbours technique better handles scaled data. This greatly affected the distribution of predicted samples since the Euclidian distance between scaled points was decreased.

Random oversampling

Random oversampling works in a similar manner to SMOTE except instead of using nearest neighbours to generate new data, it randomly selects existing data and makes identical copies of it. This is a quick way to equalise the number of samples for each feature, to mitigate the effect of having an unbalanced data set.

Classifiers:

Support Vector Machine (SVM)

SVM as a supervised machine learning algorithm is applied as a classifier to find the hyperplane in a 9-dimensional space. This is because there are 9 features which describe the glass data like the Refractive Index and 8 compositional elements. Therefore it was extremely difficult to visualise its output using a graph. As such, only scores and predictions were plotted as bar and scatter interactive graph plots. SVM was chosen as a classifier because it is effective in high dimensional spaces like in cases where the number of dimensions (9 features) exceeds the number of samples (8).[2] Furthermore, it is memory efficient, uses a subset of training points through support vectors and it is a versatile classifier where Kernel functions can be specified which is a useful tool especially when its hyperparameters are optimised.[2][3]

SVM is a good choice for classifying the glass evidence because the 9 features only exceed the 8 samples by 1 which suggests that over-fitting should not occur. If the features exceeded the samples by a lot, over-fitting would have to be avoided by choosing the right kernel.[2] The choice of the non-linear, kernel radial basis function (RBF) transforms the non-linearly separable data into an almost linearly separable data set. Sklearn.svm.SVC was used in order to build the Support Vector Machine classifier for multi-class classification with a one-versus-rest classification in order to compare a chosen type of glass to all other types. This is an effective method since each type within the data set is described by 9 different features.[3] The scaling of the data by the pre-processor class highly contributes to the reliability of the output since SVM is not scale invariant.[2][3]

Two hyperparameters were optimised in SVM with GridSearchCV 'gamma' and 'C'. While C adds a penalty for each misclassified data point, gamma defines how far the influence of a single training example reaches, selected by the model as support vectors when RBF kernel is used.[4][5] Gamma is aimed to be a low value between 0.0001 and 10 to avoid overfitting while C is aimed to be between 0.1 and 100, a large value to avoid generalized models.[4][5]

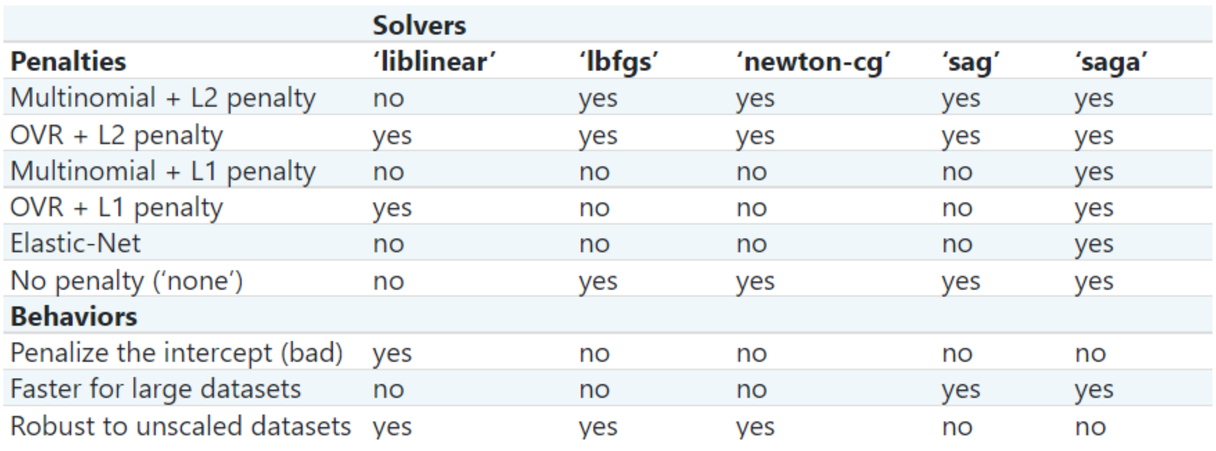
The code could be improved by deciding whether there is a need to focus on some features more than others. When optimising the hyperparameter C it is necessary to consider whether there is a need to prioritise some elemental compositions or glass types more than others with SVC’s class_weight or sample_weight parameter. This would give more importance to certain types of glass or individual samples throughout the fit method by affecting the C parameter and encouraging the classifier to get these specific samples right.[2][3]

Linear Regression (LR):

Linear regression was chosen as one of the classifiers because of its simplicity and ease of training. Furthermore, it can work with multi-class datasets through multinomial regression. However, it is prone to overfitting data especially if the number of observations is less than the number of features. This is an issue with our given dataset as the number of observations for glass type 6 was 8, which is less than the number of features totalling 9. This issue was mitigated by using oversampling during pre-processing and regularisation which is built into logistic regression.

GridsearchCV was used to fine-tune the hyperparameters used in logistic regression, namely the solver, the penalty and the C value. The choice of solver was between lbfgs, liblinear, newton-cg, saga and sag. Saga and sag were excluded because they are built to be more efficient for large-scale data, and the data we are working with is quite small. Furthermore, saga and sag have issues with convergence, likely because they are sensitive to unscaled data. Even though our data was scaled during pre-processing, there were still a lot of 0 values due to the nature of the data we were working with (elemental composition). Hence, the choice of solver was reduced to between lbfgs, liblinear and newton-cg.

Logistic regression offers built-in regularisation of data, which helps to prevent overfitting. The regularisation technique (penalty) was chosen based on the solver, since there are specific incompatibilities that must be accounted for, shown in Figure 1.



	Solvers				
Penalties	**'liblinear'**	**'lbfgs'**	**'newton-cg'**	**'sag'**	**'saga'**
Multinomial + L2 penalty	no	yes	yes	yes	yes
OVR + L2 penalty	yes	yes	yes	yes	yes
Multinomial + L1 penalty	no	no	no	no	yes
OVR + L1 penalty	yes	no	no	no	yes
Elastic-Net	no	no	no	no	yes
No penalty ('none')	no	yes	yes	yes	yes
Behaviors					
Penalize the intercept (bad)	yes	no	no	no	no
Faster for large datasets	no	no	no	yes	yes
Robust to unscaled datasets	yes	yes	yes	no	no

Figure 1. The penalties supported by each solver. The table summarises the functionalities of the solvers, the method allows the best solvers to be chosen according to the data.[6]

Finally, a sweep was done across C values, which is inversely proportional to the strength of the regularisation and is a standard parameter to optimise when using GridSearchCV. Since logistic regression assumes a linear relationship exists between the independent and dependent variables, one way to improve the reliability of using logistic regression as a classifier is to assert normal distribution. One assumption made by the regression function is multicollinearity or an established linear assumption which may not be appropriate for all data.[8] Using a normal model to fit the variables to follow a Gaussian distribution, would transform the data, similar to the transformation function in the SK-Learn pipeline.[8]

K Nearest Neighbours (KNN)

KNN was used as a technique to classify the data since the number of neighbours could be varied with an optimal value output. The weights hyperparameter was defined as ‘distance’ to ensure that closer neighbours had a bigger weight on classification, this performed better than ‘uniform’ weights since further data points may be prone to skew data and hence the label classified. The algorithm type was set to ‘auto’ to minimise human input and allow the best search method for instance, ‘ball tree’, ‘kd_tree’ and ‘brute’ to be appropriately selected for the dataset. The GridSearchCV function from scikit learn was integral in fitting and predicting the estimator as it used a dictionary of neighbours as an input, and refit the classifier using the best index and score. Initially, the cross-validation hyperparameter was selected to run five times and compare the results of each trial however, the number of splits in a particular class was too small to split successively. As a solution, the training ratio was increased to ensure the least populated class was not less than five in the data-splitting method.

KNN was a reliable classifier in producing reproducible results due to its ability to classify multi-label data with ease. In comparison with binary classifiers which rely on comparing one class with another or the rest of the classes, KNN allowed a dictionary of neighbours to classify the data which was optimised using GridSearchCV. Using the Pipeline function from scikit learn could be useful in reinforcing the split between the train set and test set however, the pre-processing hyperparameter was inappropriate for the EU, UK and US libraries given the different conventions. The method of creating a pre-processing class was therefore more robust for the performance of KNN. As an individual technique, pipeline would be appropriate however, only techniques appropriate to all models were considered.

<u>Reliability and confusion matrix:</u>

Evidently more data would improve the reliability and prediction of the code. This is a conceivable notion given machine learning requires a high volume of trials to be run in order to comment on the reproducibility of data. Although there was an overlap between classifiers, a higher volume of data would facilitate the training of each model.

A confusion matrix was used to visualise the accuracy and errors of the 3 chosen classifiers. All classifiers on the confusion matrix predict 10 to 25 samples to be Type 3 for Type 1 glasses which are false positives. An improvement would be to use ROC AUC (receiver operating characteristic area under the curve) curves which graphically portray the ability of the classifier to fit the data as a function between 0 and 1. [9] The fraction of true positives out of total positives against false positives out of total negatives would increase confidence in the fitting ability. Considering the trends in prediction, the dominant classes for LR, SVM and KNN tended to be classes 5, 6 and 7 followed by 3.

Furthermore, by using the pipeline function from SK Learn, flawed accuracy results of 0.92 and 0.96 were obtained for LR and SVC respectively. Pipeline was unsuitable for KNN as there was no transform function available. These scores were too optimistic which raised concerns about how pipeline treated different classes of data. Given the algorithm uses binary classification, it was suggested that the ‘one vs one’ multi-class classification technique used by pipeline introduced biased classification labels.

One of the biggest challenges our team has faced throughout the project was choosing whether SMOTE should be used before or after scaling the data. Since when used before the gang is “proven” to be guilty but if used after they are innocent due to fewer instances of Type 3 glass being constantly predicted. On the other hand, for the gang to be found guilty it does not matter whether Type 3 appears in a minority or majority, it just must be printed consistently on the scatter graph. The question is how the use of SMOTE before or after scaling will affect the output of false positives or true positives. Using SMOTE after scaling is technically correct since it allows the machine algorithm to be non-biased from the beginning of the investigation. Therefore, SMOTE was used after scaling since a non-biased judgement is crucial for the investigation to be fair.

Interactive plot:

The predicted scores compared to the confusion matrix displayed the difference between the accuracy and reliability of the classifiers and are presented by the interactive plots.

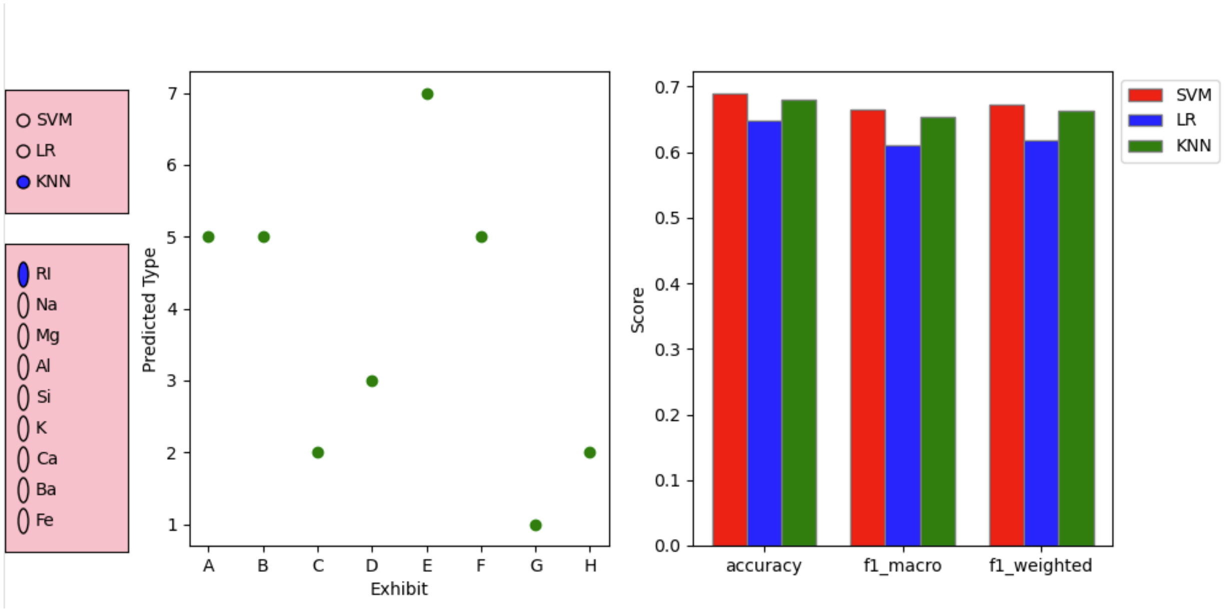


Figure 2. Interactive scatter plot and bar graph of the three classifiers. Radio Buttons allow the user to select different variables to be excluded from the pre-processing and classification classes which determine the classification labels. Once the specific feature is selected for the bar plot it will affect the distribution of the scatter plot.

Although a single classifier would suffice, the user is given results from three classification techniques due to increased robustness. The reproducibility of data was shown by trends in data such as a spread of data and evidence that Type 3 glass was present. The use of multiple models as a means of cross-validation increased the confidence of results given the limited number of exhibit samples available.

Plot 1

The scatter plot displayed the ‘Predicted Type’ as a function of the ‘Exhibit’ with the three classifiers as the radio buttons. Employing all classifiers increased the confidence in the results as there were several overlaps. Where different types were predicted, this was attributed to the ability of the classifier to compute multi-labelled data. To further increase the accuracy of prediction and the final conviction, more samples would need to be analysed as this small sample may be prone to clustering.

Plot 2

The bar chart graphed three types of scores against a numeric score for each classifier. Accuracy, f1_macro and f1_weighted were selected to display given they are universal to each classifier. F1_weighted was chosen since f1_micro showed the same result as accuracy.

Conclusion

Are they guilty? There was a consistent instance in which Type 3 glass was predicted on the scatter plot by all three classifiers, this provoked the question of whether they were a true or false positive. There is a high likelihood that they were true positives given predictions by KNN, LR and SVM. It was evident from the confusion matrix that between 24 and 26 samples were correctly labelled as Type 3 for each classifier: the fourth most populated label. Therefore, there is a confidence that the gang was culpable of the crime and is therefore liable to face charges. The consistency in score ranging between 55-70% was indicative of a well-trained algorithm handling test data with ease. The most important variable affecting classification was identified as Al due to the drastic decrease in overall score of each classifier when omitted. Factors such as the age of glass and manufacturing impurities may have introduced an element of chemical dependency although this was assumed to be negligible. To conclude, whilst the result of this investigation should not wholly depend on the code's result as the samples are randomly selected and shuffled, there is strong evidence condemning the gang time and time again.

Resources:

[1] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, *Journal of Artificial Intelligence Research* **2002**, *16*, 321.

[2] Scikit – learn: Machine Learning in Python, Scikit – learn developers, 1.4 Support Vector Machines, URL: 1.4. Support Vector Machines — scikit-learn 1.2.0 documentation, accessed: January **2023**.

[3] GeeksforGeeks, aswathisasidharan, Support Vector Machine Algorithm, URL: Support Vector Machine Algorithm - GeeksforGeeks, accessed: January **2023**.

[4] Towards Data Science, Soner Yildrim, Hyperparameter Tuning for Support Vector Machines – C and Gamma Parameters, URL: Hyperparameter Tuning for Support Vector Machines — C and Gamma Parameters | by Soner Yıldırım | Towards Data Science, accessed: January, **2023**.

[5] Scikit – learn: Machine Learning in Python, Scikit – learn, RBF SVM parameters, URL: RBF SVM parameters — scikit-learn 1.2.0 documentation, accessed: January, **2023**.

[6] Scikit – learn: Machine Learning in Python, Scikit – learn, Linear Models, URL: 1.1. Linear Models — scikit-learn 1.2.0 documentation, accessed: January **2023**.

[7] Scikit – learn: Machine Learning in Python, Scikit – learn, f1_score, URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html#sklearn.metrics.f1_score, accessed: January, **2023**.

[8] Keboola, The Ultimate Guide to Logistic Regression for Machine, URL: https://www.keboola.com/blog/logistic-regression-machine-learning, accessed: January **2023**.

[9] Hajian-Tilaki, Caspian J Intern Med, Receiver Operating Characteristic (ROC) Curve Analysis for Medical Diagnostic Test Evaluation, URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3755824/ , accessed: January **2023**.

[10] Y. Verma, “Why Data Scaling is important in Machine Learning & How to effectively do it,” URL: Why Data Scaling is important in Machine Learning & How to effectively do it, **2021**.

[11] A. Kumar, “MinMaxScaler vs StandardScaler - Python Examples,” can be found under MinMaxScaler vs StandardScaler – Python Examples, **2020**.