Computing Challenge 22-23 – Aidan, Daniel, Aaron, Gabriel

Code Report and Brief Summary

In the challenge, the aim is to use one machine learning model to predict types of glass in the evidence samples of a crime scene. This is achieved by training the candidate classifiers with US, EU and UK databases for 7 types of glasses, to find the winner of highest predictivity. The target is to determine whether type 3 glass is present in the evidence or not as the suspects used it in the crime. Therefore, suspects can be convicted.

The body of the code is separated into mainly 4 parts: A, B and C, with part D being the final prediction on the evidence sample. Detailed description of specific codes are shown as markdowns or '#' lines in the code file.

Starting with part A, 3 classifiers are chosen: decision tree, random forest, and k nearest neighbours. Other models like support vector machine and regression models are not considered due to their complexity and less capability in multi-classification problem (in this case, 7 categories). Then, 3 database files used to train the models are processed: replacement of NaN value with mean and scaling with standard scaler. RandomOverSampler[1] is used to over sample the minority features and ensure balanced representation of each feature. Both the balanced and original data are split into training and test data sets with ratio 8:2 using train_test_split[2] method in scikit learn. The 3 classifiers are then trained and tested.

In part B, to reduce potential effect of overfitting by only using one train/test split in part A, 5-fold cross validation are conducted for the models and less biased results of accuracies are displayed. 5-fold is used because of the relatively small data set. So far only default hyperparameters are used for each model. To find optimum hyperparameters, RandomSearchCV method[3] is selected to test out random combinations of values for hyperparameters, whose ranges are suggested as input. The ranges of values for any hyperparameter is chosen rather arbitrarily but to also cover as many combinations as possible. Note that for random forest, the tuning using both original and balanced data requires around 1 hour to complete. This is because possible combinations of hyperparameters are significantly more. 'n_iter' of RandomSearchCV for random forest is set to be 5000 compared to 3-digit numbers for other two models (details of RandomSearchCV implementation are in the code). With tuned classifiers, they are put to the final prediction of the test set and 5-fold cross validated to show the accuracies. Random forest is found to have best predictivity.

Note that in both part A and B, both original and balanced data are used during the testing. This is nothing but to fully illustrate that training in balanced data results in better predictions in every step, before and after hyperparameter tuning, or through test set prediction and cross validation.

In part C, an interactive graph using library Plotly[4][5][6] is made to demonstrate effect of single feature on accuracy of the model. In this part, all 3 models are used. Detailed description is in the code file.

For the final part D, the winner random forest with tuned hyperparameters determined in part B predicts the types of glass samples in the evidence. Type 3 glass is found with high precision. In conclusion, the suspects can be convicted.

Reference (Python Functions/Libraries Used Outside Lecture Notes)

[1] ‘from imblearn.over_sampling import RandomOverSampler’ was used.

Can be found at ‘https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.RandomOverSampler.html’

[2] ‘from sklearn.model_selection import train_test_split’ was used.

Can be found at ‘https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html’

[3] ‘from sklearn.model_selection import RandomizedSearchCV’ was used.

Can be found at ‘https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html’

[4] ‘import plotly.express as px’ was used.

Can be found at ‘https://plotly.com/python/plotly-express/’

[5] ‘import plotly.io as pio’ was used.

Can be found at ‘https://plotly.com/python/renderers/’

[6] ‘import plotly.graph_objects as go’ were used.

Can be found at ‘https://plotly.com/python/graph-objects/’