SPRINGBOARD DATA SCIENCE CAREER TRACK

CAPSTONE PROJECT #1 – PREDICTIVE MODEL

SILVIA MAIONE

The goal of this project is to be able to predict a tree's health (fair, good or poor), from the knowledge of some data such as the diameter and any problems. This goal makes it a multi-class classification problem. The algorithms taken into consideration are Random Forest and KNN (k-Nearest Neighbors). Due to the findings from the data wrangling and statistical analyses, the model is trained and tested separately for the five boroughs and using certain features, but a classifier is also trained with data from all boroughs for comparison purposes.

For each borough, the dataset is split into training and test sets. The test set is set aside as a hold-out set, to be used for final validation. Since most of the labels are in one class (the good health), the training set is imbalanced. Two samplers available in the "imblearn" package and a custom one are used and compared.

Precision, recall, and f1-score are the metrics used for evaluation and comparison. For the minority classes, recall is particularly important, and vice versa, for the majority class, high precision is desirable. In fact, a high number of false negatives in the formers and a high number of false positive in the latter would mean to classify some trees in fair or poor shape as in good health. ROC curves are also plotted for each class and AUC calculated.

A first "raw" comparison is made between the models "out-of-the-box", with no cross-validation, nor tuning or balancing. As expected, the performances aren't great. Next, the training set is used for cross validation. Both Random Forest and KNN give very high scores on training during cross-validation, but the scores on the hold-out set isn't good. The models are probably overfitting. For this reason, PCA is used to plot the explained variance of the features, showing that one feature, most likely the tree diameter, is responsible for most of the variance. So, additional feature analysis is done, with both SelectKbest and RFE, and it confirms that the tree diameter is the main feature, while others are less important. With these

results in mind, a grid search is conducted to tune the hyperparameters. Datasets containing both the tree diameter and only one other feature are used. The grid search functions available in Python use averaged scoring, and for multi-class classification problems as this one, only some of the various type of averaging is available (“macro” and “weighted”). Also, ideally a search would be done to maximize different scores depending on the class, as mentioned earlier, but to the author’s knowledge this can’t be done with the available grid search tools. It would require a custom search looping through various combinations of the parameters. For these reasons, the f1 score is used as a scoring function in the grid search, specifically the macro average score, in order to give the same importance to all classes (with a micro average score, the largest class would have more influence).

Depending on what samplers and features are used, the recall of one of the minority class improves, but the recall of the other minority class gets worse. For the final classifier, all the features are used again, since this seems to give slightly better results. Overall, the results aren’t good, mostly because the scores of the minority classes are very low. Random Forest gives better results on Manhattan data, so it used for the final classifiers of the other boroughs too. The majority class has 80-90% precision, while for the other two classes recall is about 25-50%, depending on the borough and the class. These scores are on a hold-out set, put aside with the initial split of the data.

The classifier based on all boroughs data performs similarly to the separate boroughs model.