Feature selection

Feature selection is an important step when dealing with datasets having a considerable amount of variables. There exist a number of readily available functions in sklearn which can be implemented to achieve automatic feature selection. Feature selection methods can be split in four main categories.

These are the removal of features with low variance, univariate feature selection, recursive feature elimination and feature selection using SelectFromModel. For our dataset we used one method from the univariate feature selection and using SelectFromModel.

Feature removal which have low variance is a simple and straightforward method to implement and is considered to be the baseline approach for feature selection. The method used in this case is VarianceThreshold and by default, it removes all the features having the same value in all samples.

The second category for feature selection is univariate feature selection and scikit-learn implements three functions namely SelectKBest, SelectPercentile and GenericUnivariateSelect. SelectPercentile was chosen from this category as it removes all but a user-specified highest scoring percentage of features using common univariate statistical tests for each feature. The selection is made based on a percentage of the original features.

The third category being recursive feature elimination is given an external estimator which assigns weights to features and recursively selecting features shrinking the feature sets each time. The estimator is trained on an initial set of features and the least important features are pruned from the current set of features. This step is repeated until the desired number of features is met.

The fourth category is using the SelectFromModel which was also used on the dataset. This method utilises a supervised model to determine the importance of each feature. This model has a featureimportances attribute after fitting and features are not considered important and are thus removed if the featureimportances values fall below the provided threshold parameter.

Cross validation

Training and testing sets are picked and separated when building a model. This can be done using popular library methods which will split the data in these two sets. This allows for the calculation of the testing accuracy of the model on out-of-sample data. Cross-validation is an improvement to this model in such a way that essentially, a number of training and testing splits are done. This gives the benefit of calculating the testing accuracy for each of these splits hence reducing the variance present in the model.

The process of cross-validation can be split in a few steps. Step 1 is to split the data in k folds. The value of k can be decided arbitrarily although there are best practices to choose optimal values for it. The next step would be to take fold 1 as the test set and the union of the other folds as the training set. Having split the data, the next step would be to calculate the testing set accuracy. This process would be repeated for k times using a different fold each time for the testing set. The test set accuracy would be calculated by averaging all the obtained test set accuracy results from all the different folds. This would give a better estimate for the out-of-sample accuracy.

Cross-validation is better than a single training and testing split because it generates a more accurate accuracy result. It also utilises the data more efficiently by utilising every observation for training and testing. Naturally, training and testing splits runs k times faster than cross-validation. It is also simpler to examine the detailed results of the testing process as well.

There are best practices for choosing the best parameter for k in cross-validation. The said parameter can be any number, however a value of 10 is generally recommended. Other improvements which could improve cross-validation would be to repeat the cross-validation process with different random splits of the data. This would essentially produce a more reliable estimate of the out-of-sample performance by reducing the variance linked with a single instance of cross-validation.