INFS4203/7203 Project
Semester 2, 2022
Phase1: project proposal

Shuo Yuan Author
School of Information Technology and Electrical Engineering
The University of Queensland, Qld., 4072, Australia

1 Data pre-processing

Data pre-processing can refer to manipulation or dropping of data before it is used in order to ensure or enhance performance[1]. Analysing data that has not been carefully screened for such problems can produce misleading results. Thus, the representation and quality of data is first and foremost before running any analysis[2]. Knowledge discovery in the training phase is more difficult if there is a lot of irrelevant and redundant information or if there is noisy and unreliable data. Moreover, data pre-processing may affect the way to interpret the final data processing results.

Since we may choose different data pre-processing techniques to achieve the best performance for different datasets, we should first observe and analyse the given "Ecoli.csv" file. With the help of the "pandas" library, we can find that the first 103 columns in this dataset have at least 80 missing values each, and more often have about 100 missing values, and the appearance of these missing values is not regular. If we choose to delete all rows with missing values, it is likely that the dataset will end up with few rows of data left. Therefore, it is better to use imputation to handle them.

After that, we should consider the impact of outliers on the classification. Since in data analysis, outlier detection is generally understood to be the identification of rare items, events or observations which deviate significantly from the majority of the data and do not conform to a well-defined notion of normal behaviour[3]. Detecting outliers and eliminating them will have a positive effect on our overall classification model.

We also note that the range of values between the different columns is not the same and differs greatly from each other, for example, Column2 takes the range [-125.16, 118.60], while Column5 takes the range [-9.20, 13.03]. Since we cannot guarantee that the degree of magnitudes is the same between different columns, and the magnitudes may impact the classification! In order to eliminate such a problem, we need to perform data normalization to solve the comparability between different columns.

The next few subsections list the pros and cons of different data pre-processing techniques.

1.1 Outlier detection

In this proposal, we will only discuss the following methods[4]:

1) Density-based technique: Objects are evaluated with respect to their density. The smaller the density is, the more likely this point is an anomaly. Good at detecting outlier from data with variations in density but it is difficult to determine k and need to calculate the distance between all data points. sklearn.neighbors.LocalOutlierFactor() function would help us to implement it.
2) Model-based technique: A distribution model is assumed on the data, most widely used is gaussian distribution, and objects are evaluated with respect to how well they fit the model. But it is hard to determine the distribution model. sklearn.covariance.ELLipticEnvelope() function would help us to implement it.
3) Distance-based technique: Objects are evaluated with respect to their distances to their k-th nearest neighbour. Simple and straightforward but difficult to determine k and will face low efficiency and curse of dimensionality. Less used nowadays.
4) Cluster-based technique: Apply clustering to the data and objects are evaluated with respect to how strong they belong to each cluster. The hyperparameter is hard to setting for clustering algorithm and outlier may distort the clustering. Also face low efficiency and curse of dimensionality.
5) Isolation-based technique: Split the data until all points are isolated. Since outlier will be isolated in the early stage of partition, we could easily detect them. Efficient in both computational and memory, but some unreliable results may be generated due to each partition only involves one feature. Sklearn.ensemble.IsolationForest() function would help us to implement it.

In this project, we only implement the Density-based, Model-based and Isolation-based techniques because it is hard for us to determine the

hyperparameter k in Distance-based and Cluster-based techniques.

1.2 Imputation methods

1) Imputation by feature's all values: for numerical value: imputation the missing numerical value by the average of the feature's all values; for nominal value: imputation the missing nominal value by the most common feature value.
2) Imputation by feature's class-specific values: for numerical value: imputation the missing numerical value by the average of the feature's class-specific values; for nominal value: imputation the missing nominal value by the most common class-specific feature value[5].

1.3 Normalization methods

1) Max-min normalization: Get all the scaled data in the range (0, 1). Min-max normalization preserves the relationships among the original data values. Be helpful when data does not follow a Gaussian distribution and be useful with algorithms that do not assume any distribution of data.
2) Standardization (z-score normalization): A variation of scaling that represents the number of standard deviations away from the mean. Be useful when data follows a Gaussian distribution and there are a few outliers, but not so extreme that you need clipping[6].

1.4 Select appropriate techniques with cross-validation

All of the above pre-processing methods have their own suitable scenarios, pros and cons, but there is no way to tell at a glance which method is the most suitable for "Ecoli.csv" dataset, especially when it has so many features and missing values. The pre-processing techniques are sequentially divided into 3 steps: Imputation → Outlier detection →Normalization. Actually, this is a combination problem like which pre-processing technique should be combined with which classification method. Therefore, we need to use cross-validation (a tool for evaluating any technique) to filter out the best one. The datasets obtained from different combination of different pre-processing techniques and classification method are written into individual CSV files, which would be used as input for cross-validation to obtain the respective validation score. But here comes exploding combinations problem, we need to empirically reduce the combinations[7]. Finally, the combination with the highest score (The score assessment methodology will be described in detail in the third paragraph) will be selected.

2 Apply classification techniques

In this project, we are going to apply four classification techniques learned in lectures (decision tree, random forest, k-nearest neighbour and naïve bayes) to the above pre-processing data. In order to save time and effort and implement these four classification techniques correctly, we are going to use the functions already written in the "sklearn" package, because they already have the implementation process ready and quite mature, we only need to focus on tuning the hyperparameter instead of implementing the whole algorithm ourselves.

In the next few subsections, I will introduce in turn the functions in sklearn for these four classification techniques and their key hyperparameters that we need to tune ourselves. For parameters that need to be set as integers or float numbers, such as max_depth in the decision tree, we will first tune in a large interval and then narrow down the range step by step. And for other parameters to be specified by name, we will try the possible values one by one, such as criterion in the decision tree, and finally use cross-validation to filter out the best hyperparameters.

2.1 Decision tree

sklearn.Tree.DecisionTreeClassifier() would construct a decision tree classifier. The hyperparameters we care most about in this function are:

criterion: The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "log_loss" and "entropy" both for the Shannon information gain.

max_depth: The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples. By setting this integer or float parameter, you can end the tree growth in advance to improve the over-fitting problem.

min_samples_split: The minimum number of samples required to split an internal node; default is 2.

min_samples_leaf: The minimum number of samples required to be at a leaf node. This may have the effect of smoothing the model, especially in regression. Default is 1.

We can set the values of the above hyperparameters to achieve the pruning, because it's not necessarily to always use all the features.

2.2 Random Forest

sklearn.ensemble.RandomForestClassifier() would construct a random forest classifier. The hyperparameters we care most about in this function are:

n_estimators: The number of trees in the forest; default is 100.

criterion: measure the quality of a split. Could select "gini", "log_loss" and "entropy".

max_depth: The maximum depth of the tree.

max_features: The number of features to consider when looking for the best split: If int, then consider max_features feature at each split. Choosing a suitable value for it can prevent overfitting and greatly improve the generalization ability of the model.

bootstrap: Whether bootstrap samples are used when building trees.

max_samples: If bootstrap is True, the number of samples to draw from X to train each base estimator. If int, then draw max_samples samples.

2.3 K-nearest neighbour

sklearn.neighbors.KNeighborsClassifier() would implement the k-nearest neighbors vote. The hyperparameters we care most about in this function are:

n_neighbors: Number of neighbors to use by default for kneighbors queries; default is 5.

algorithm: Algorithm used to compute the nearest neighbors: 'ball_tree' will use BallTree, 'kd_tree' will use KDTree, 'brute' will use a brute-force search.

p: Power parameter for the Minkowski metric. When p = 1, this is equivalent to using manhattan_distance (L1), and euclidean_distance (L2) for p = 2. For arbitrary p, minkowski_distance (L_p) is used.

2.4 Naïve Bayes

Sklearn provides three functions to deal with different data distributions:

1) sklearn.naive_bayes.GaussianNB() implements Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian.
2) sklearn.naive_bayes.MultinomialNB() implements the naive Bayes algorithm for multinomially distributed data
3) sklearn.naive_bayes.ComplementNB() implements the complement naive Bayes (CNB) algorithm.

In general, GaussianNB should be used for features in decimal form. GNB assumes features to follow a normal distribution. MultinomialNB should be used for the features with discrete values like word count 1,2,3... BernoulliNB should be used for features with binary or boolean values like True/False or 0/1[8]. Since we assume that "Ecoli.csv" follows normal distribution, so we would only use GaussianNB function.

The hyperparameters we care most about GaussianNB are:

priors: Prior probabilities of the classes. If specified, the priors are not adjusted according to the data.

var_smoothing: Portion of the largest variance of all features that is added to variances for calculation stability; default is $1e^{-9}$.

2.5 Ensemble methods

The goal of ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability/robustness over a single estimator[9]. After finding the best cross-validation score for each of the above four classification techniques, we will use ensemble methods to try to see if we can come up with a better cross-validation score. For example, ensemble of one KNN classifier and a random forest classifier or ensemble of four decision trees trained on subsets of different features.

3 Evaluate the model

Confusion Matrix:

	Predicted Positive	Predicted Negative
Ground Truth Positive	TP	FN
Ground Truth Negative	FP	TN

Based on the Confusion Matrix, there are two criteria to evaluate the model:

1) Accuracy = (TP + TN) / (TP + FP + FN + TN)
2) F1-scores:
   Precision = TP / (TP + FP)
   Recall = TP / (TP + FN)
   F1-scores = 2 / ((1/Precision) + (1/Recall))

But when we look back at the "Ecoli.csv" data set, we find that 1339 samples of the total 1500 samples are negative and only 161 samples are positive. The positive class has low proportions in the data as compared to the negative class. Recall from the lecture that accuracy may not be the most appropriate metric if the data is imbalanced[10], because if a model predicts all samples as negative, then the accuracy will also look high, but in fact this does not help in classification, and we cannot get any valid information from this model.

Finally, we will use cross-validation to evaluate the models, and the metric will be F1-scores. At the end of cross-validation, we will get the validation results in each iteration, and then we choose the best model with the highest mean F1-scores and the smallest possible standard deviation.

4 Timeline

Week 8: Complete data pre-processing part, use cross-validation to filter out the pre-processing methods and classifier hyperparameters that lead to the highest average validation f1-score.

Week 9: Construct the four classification methods mentioned in lecture and ensemble methods. Using cross-validation to choose the best hyperparameters for each method.

Week 10: Apply these classification algorithms to our split test set with cross-validation, choose the best generalization one as our optimal classification model.

Week 11: Using the test set given by teaching team to test and obtain the final f1-score.

Week 12: Write the project report according to the experimental result.

References

[1] "Guide To Data Cleaning: Definition, Benefits, Components, And How To Clean Your Data". Tableau. Retrieved 2021-10-17.

[2] Pyle, D., 1999. Data Preparation for Data Mining. Morgan Kaufmann Publishers, Los Altos, California.

[3] Chandola, V.; Banerjee, A.; Kumar, V. (2009). "Anomaly detection: A survey". ACM Computing Surveys.

[4] Miao Xu. "Lecture 7: Anomaly Detection". University of Queensland, Data Mining (INFS4203/7203), pp. 14-56, 2022.

[5] Miao Xu. "Lecture 2: Introduction to Classification". University of Queensland, Data Mining (INFS4203/7203), pp. 46-50, 2022

[6] Miao Xu. "Lecture 2: Introduction to Classification". University of Queensland, Data Mining (INFS4203/7203), pp. 52-55, 2022

[7] Hao Yang. Ed — Digital Learning Platform. (n.d.). Retrieved September 14, 2022, from https://edstem.org/au/courses/9008/discussion/1012972

[8] Bernoulli NB vs MultiNomial NB, How to choose among different NB algorithms? (2017, January 27). Cross Validated. Retrieved September 15, 2022, from https://stats.stackexchange.com/questions/258458/bernoulli-nb-vs-multinomial-nb-how-to-choose-among-different-nb-algorithms

[9] 1.11. Ensemble methods. (n.d.). Scikit-learn. Retrieved September 14, 2022, from https://scikit-learn.org/stable/modules/ensemble.html#ensemble

[10] Miao Xu. "Lecture 2: Introduction to Classification". University of Queensland, Data Mining (INFS4203/7203), pp. 28-36, 2022