Preface

In this document, we conducted an experiment to find the appropriate hyperparameters to form two algorithms, k-Nearest Neighbour (kNN) and Decision Tree (DT) algorithms, based on the provided meta dataset. After analyzing how each model fared, we used the kNN model to generate a result from the meta dataset, before comparing how each model fared.

February 2023

Organization

Team Members:

Song Zijin	National University of Singapore, Singapore
Nauman Sajid	National University of Singapore, Singapore
Lim Zhe Kang	National University of Singapore, Singapore

Table of contents

The search for hyperparameters of k-Nearest Neighbour (kNN)

The final choice of hyperparameters to use:
n_neighbors = 51, weights = 'distance',metric = 'minkowski'

The experiment:

1. The dataset is divided into two groups, the train and test dataset, with the train-test-split function
2. Using GridSearchCV, the train data is passed through kNN classifiers of different hyperparameters, using a 5-fold cross-validation, with the entire list of parameters used shown below:

```
a. 'n_neighbors' : list(range(3, 70)),
b. 'weights' : ['uniform','distance'],
c. 'metric' : ['minkowski','euclidean','manhattan', 'haversine',
   'cosine']
```

3. After this pass, the results are exported to an excel file (KNN test results.xlsx) and evaluated
4. Since the 194 cases with the highest score are all use ‘distance’ as weight, we conclude that ‘distance’ is the better hyperparameter for this category
5. A more specific test is conducted using 4 combinations of different matrices with the highest mean score, using a median k value for the combinations with the same method.
6. These combinations are passed through stratified 10-fold cross-validation for the training set, and the trained model is used on the testing set
7. The result revealed that both ‘minkowski’ and ‘euclidean’ share the same accuracy, and are the models with the highest accuracy for both the training set and testing set, with a training set accuracy of 64.7% and a testing set accuracy of 80.7%.
8. Another round of fine-tuning is conducted where the method is fixed, while the k value uses the top 4 highest scoring ones associated with ‘minkowski’ in the initial test. The k value, 51, with the highest accuracy among them is used

The experimental results:

The set of hyperparameters that was chosen in the end had an accuracy of 64.8% on the training dataset and an accuracy of 80.7% on the testing dataset. More details of the accuracy of other combinations of parameters can be found in the excel file or the Jupital Notebook file named kNN.

Evaluating with other scoring matrices:

Other than accuracy, the f1_micro is also used as a scoring matrix for another round of GridSearchCV, with results stored in kNN test results v2. The best hyperparameters

according to the new scoring and from the previous experiment are compared using a confusion matrix, with the parameter found in the previous experiment still remaining as best.

The search for hyperparameters of Decision Tree (DT)

The final choice of hyperparameters to use:
criterion=entropy, max_depth=2

The experiment:

1. The dataset is divided into two groups, the train and test dataset, with the train-test-split function
2. First RandomizedSearchCV is used to approximate the best hyper parameters on train set based on accuracy and f-scores metric
3. Then grid search is used around the range of the best parameters found in step 2 to try and find the local maximum in the region found by the RandomizedSearch

```
params = {
    'max_depth': [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 20, None],
    'criterion': ["gini", "entropy"],
    'min_samples_split': [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 50],
    'class_weight' : ["balanced",None],
}
```

4. F_scores metric produced had overfitting issues (train accuracy of 85% while test accuracy of 58%) and so pruning was done on the decision tree to find a better model

The experimental results:
For accuracy the chosen hyperparameters produced a test accuracy of 76% and train accuracy of 66%. - (criterion=entropy, max_depth=2).
For F_scores the chosen hyperparameters produced a test accuracy of 64% and a train accuracy of 68%. - (criterion=entropy, max_depth=7, min_sample_splits=5, ccp_alpha=0.0445931).

Evaluating:
For accuracy hyperparameters the confusion matrix showed high accuracy for majority classes (class 1) but poor accuracy for other classes (class 0 and class 2) because it seemed as though the model was biased towards class 1. Hence weighted f1_scores were used as scoring metrics to derive other hyper parameters while accounting for number of instances in each class. F1 scores produced a confusion matrix that was less skewed in favor of the majority class than the one generated based on the best hyperparameters based on accuracy. However, the new confusion matrix better predicted instances of class 2 only and class 0 was as bad as before while class 1 accuracy deteriorated. Moreover, there was a reduction in overall test accuracy from 76% to 68%. Hence, hyperparameters (criterion=entropy, max_depth=2) were chosen.

The choice of h^*_D

In the end, we have chosen to use the kNN model with parameters: n_neighbors = 51, weights = 'distance', metric = 'minkowski', due to the fact that it has the best performance under the various tests.

Generating h'_D

Steps:

1. Convert D to a set of meta-features, this is done using the Distance-Weighted Class-Homogeneous Neighbourhood Ratio (Chen et. al, 2020) to calculate the b values.
2. Pass the converted D into h^*_D to get a classification
3. The result of the classification is 1, suggesting that KNN with default hyperparameters is better.

Comparing h^*_D and h'_D

Using the same training and tests derived from D that is used on h^*_D, we first trained h'd. Similar to h^*_D, h'_D is evaluated using a 10-stratified-fold with the same random seed. Afterwards, the test data is passed into h'd to get a resulting classification. The resulting classification is then tested with the actual classification using the Accuracy test.
The results are as follows:

	h^*_D	h'_D
Average Cross-validation score	0.6478676470588236	0.5827573529411765
Accuracy score	0.8070175438596491	0.631578947368421

As seen from the table above, h^*_D has significantly higher values for both the Average Cross-validation score and accuracy score.
There are several possible reasons for this improved performance for h^*_D.
For example, it could be because the hyperparameters of the h^*_D model were better optimized as grid search is used. It is also possible that it is due to h*d having a higher value of k, which means that h^*_D relied more on the overall structure of D, whereas h'_D relied more on the local structure. As a result, Model h^*_D was more robust to outliers and noise, and better adapted to the general structure of D, resulting in a better 10-fold cross-validation and accuracy score compared to h'_D.

In conclusion, the comparison suggests that h^*_D is the superior model. However, further experimentation can still be done to strengthen this conclusion. For instance, more metrics such as f1_score, confusion matrix and AUC-ROC can be used to sieve out any biases in D

so that a more fair comparison is done. K stratified fold can also be run multiple times with different random seeds each time to provide a more robust estimate.