CSI 5155 Assignment 2 Report

Name: Zhikun (Sam) Yuen Student ID: 300323972

Question 7

Step 1:

Accuracy table (algorithms vs datasets):

	DT	RF	SVM	KNN	MLP	GB
dataset_d	0.9735	0.9804	0.9804	0.9804	0.9804	0.9804
DB1	0.9549	0.9730	0.8600	0.8886	0.9428	0.9735
DB2	0.5348	0.6542	0.6694	0.6417	0.6512	0.6022
labor-relations	0.9133	0.9267	0.9267	0.9067	0.7367	0.9467
heart-disease	0.8043	0.8247	0.8349	0.8484	0.8182	0.8451

Table 1: Accuracy of the 6 algorithms against the 5 datasets (60 iteration in random search)

Step 2: Rank each algorithms in each dataset and compute the average rank for each algorithm. Also, computing $\overline{R}$ the sum of squared difference $\sum_j (R_{ij} - \overline{R})^2$.

	DT	RF	SVM	KNN	MLP	GB
dataset_d	6.0	1.0	1.0	1.0	1.0	1.0
DB1	3.0	2.0	6.0	5.0	4.0	1.0
DB2	6.0	2.0	1.0	4.0	3.0	5.0
labor-relations	4.0	2.0	2.0	5.0	6.0	1.0
heart-disease	6.0	4.0	3.0	1.0	5.0	2.0

$k = 6$ and $n = 5$

$\bar{R} = \frac{6+1}{2} = 3.5$

Step 3: compute average ranking for each algorithm

	DT	RF	SVM	KNN	MLP	GB
avg_rank	5.0	2.2	2.6	3.2	3.8	2.0

Step 4:

$k = 6$ and $n = 5$

$\bar{R} = \frac{6+1}{2} = 3.5$

The sum of squared differences

$n \sum_j (R_j - \bar{R})^2 = 5 \times (1.5^2 + 1.3^2 + 0.9^2 + 0.3^2 + 0.3^2 + 1.5^2) = 35.9$

The sum of squared differences $\frac{1}{n(k-1)} \sum_{ij} (R_{ij} - \bar{R})^2 = \frac{1}{25} \times 109.5 = 4.38$

Friedman statistic = $\frac{35.9}{4.38} = 8.20$

The critical value is 10.49 if $\alpha = 0.05$

Since abs(Friedman statistic) < 10.49, there is not significant difference between the 6 algorithms.

Question 8

In this assignment, I choose label "**Caff**" from the last assignment to construct dataset D because the average accuracy of the 4 models on it are the highest (0.9743). As most of the labels are imbalance, this means if the accuracy is higher, the label may be more imbalance. In "Caff", only 37 data are in class 0 (Non user), the rest of 1848 data are in class 1 (User). This dataset is extremely imbalance.

For all 5 dataset, I use the same flow to process the data, normalizing both numeric and ordinal data with MinMaxScaler() to [0,1], turn nominal features into one-hot vectors. If there are missing value in the data, I fill the missing numeric features with mean, ordinal and nominal features with most frequent value.

For all 6 models in all 5 datasets, I just random search to fine tune the model's parameters with highest accuracy. Since the RandomizedSearchCV in sklearn will search with cross validation, I train a new pipeline/model with the best parameters again after hyper-parameter search. I do 10-fold cross validation for that model and compute the average test accuracy among the 10-fold testing data. The average test accuracy is used to construct Table 1 in Q7. For dataset D, labor-relations and heart-disease, I just use cross_validation() function in sklearn to do 10-fold cross validation automatically and use the average 10-fold test accuracy as the result. For dataset DB1 and dataset DB2, I use KFold() in sklearn to create 10-fold train and test data. In each train fold, I do SMOTE oversampling or RandomUnderSampler. For the test folds, I just test them as usual.

Since dataset D is highly imbalance, I try SMOTE to oversample and RandomUnderSampler to undersample. These two sampling approaches are implemented by "imblearn" package. SMOTE is a very popular oversampling approach to generate new samples for the minority class (increase the size of minority to be the same as majority class). RandomUnderSampler just randomly picks majority samples to undersample the majority class to be the same size as minority. In dataset D, algorithms can achieve very high accuracy because the model predicts almost the majority class label only. If we use oversampling or undersampling to make the data become balance, the accuracy on the balance datasets will drop because the models do not predict the majority class only for a balance dataset.

If we use SMOTE oversampling to make dataset DB1, some algorithms' accuracy drops slightly, e.g. decision tree, random forest, MLP and gradient boosting. Their average accuracy is still higher than 0.90 after oversampling the data. The result is very good for a balance dataset. Other 2 algorithms' accuracy drops to > 0.85 (still very good). Algorithms' accuracy drops after oversampling is because they do not predict the majority class only for the balance dataset. This shows that SMOTE oversampling is more suitable for random forest, decision tree and MLP in this dataset DB1. Also, **SMOTE oversampling can really help address the data imbalance problem**.

The data is highly imbalance, if we do undersampling, a lot of information of the dataset will lost. There are only 37 minority class data. After doing RandomUnderSampler in each training fold, the data point of minority class will be even smaller than 37. If we do random sampling, most of the majority class data are dropped. Lots of information of the original dataset/majority class is gone.

We can see the accuracy of all algorithms drops a lot for dataset DB2. Most of their accuracy is about arount 0.6x. This is because the minority class is too extreme. Decision tree has the lowest accuracy (nearly 0.5, random guess), no matter how I find tune the hyper-parameters of it. Although the data imbalance problem is solved, lots of information of the majority class is lots after doing random undersampling. Also, the models' performance is not good in this undersampling dataset DB2. This shows that **random undersampling approach is not suitable for some datasets if its minority class is very small**.

Actually, **MLP is very difficult to tune the hyper-parameters in dataset DB2, Labor-relations and heart-disease**. If I just set 30 iterations for random search. MLP's accuracy is much lower than doing 60 iterations. Sometimes, the accuracy of MLP in DB2 is only 0.4 - 0.5. I need to add more iterations for random search, the MLP has a very good results (0.94 in oversampling data). This case is not shown in other models. Most of them give a very similar accuracy even though I add more iterations for random search. MLP is harder to fine-tune and hyper-parameters affects a lot for the accuracy. However, **doing 10-fold cross-validation with random search is very time consuming for large iteration number**.

In labor-relations, there are missing values, I use the mean to fill the numeric features and most frequent values to fill the categorical features.

For the labor-relations and heart-disease datasets, they are more balance than dataset D. If we just look at the accuracy metric, it is more reliable. Only MLP achieves much lower accuracy than the others especially in labor-relations data. It is because MLP is very hard to fine-tune. Other algorithms works similar in different datasets.

Gradient boosting has the best average ranking among the 5 dataset and decision is the worse.