Anushka Sinha
02718 HW4
11/12/2023

1. **Data preprocessing pipeline ("datasets/train.xlsx"):**

1.1:Fill in the Patient ID column

in the jupyter notebook (preprocess.ipynb)

1.2: Convert sex to integer encoding: 0—female, 1—male

in the jupyter notebook (preprocess.ipynb)

1.3: Convert datetime to the string format.

in the jupyter notebook (preprocess.ipynb)

1.4: Exclude the patients with missing labels (e.g., missing ID, missing timepoints).

in the jupyter notebook (preprocess.ipynb)

1.5: Calculate the number of days (LOS) each patient stays in the hospital

in the jupyter notebook (preprocess.ipynb)

1.6: Set negative LOS values to zero.

in the jupyter notebook (preprocess.ipynb)

1.7: Drop the features whose values are least informative (e.g., all the same, all NaN values)

in the jupyter notebook (preprocess.ipynb)

1.8: For the current dataset, each patient has multiple records. Compress the records to one row for each patient. If there are multiple values, choose the median.

in the jupyter notebook (preprocess.ipynb)

1.9: Remove features with more than 10% missing values.

in the jupyter notebook (preprocess.ipynb)

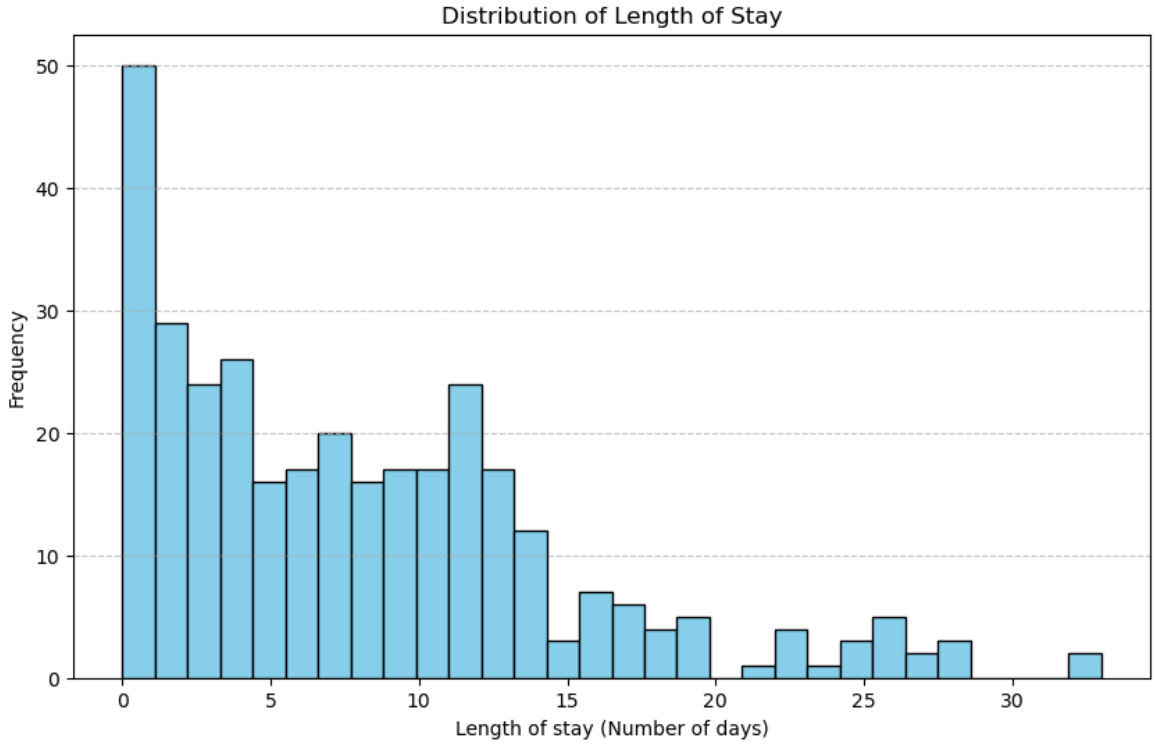
1.10 Provide the name of the method you would like to choose here:

Imputation method used here is replacing missing values with median.
in the jupyter notebook (preprocess.ipynb)

1.11: What other data preprocessing steps can you come up with? Implement them and explain why you use them here.

Since we have already filtered out features that have low variance and only keep relevant features and have removed features with more than 10% missing values, the only step left was to normalize the features, except for the labels (LOS, Outcome). This helps in reducing computing time and power when training and testing the various models. I used sklearn's StandardScaler.

1.12: Plot your final LOS distribution of the processed dataset using a histogram here.



2. Experimental Settings:

2.1: Use 4 algorithms to construct models for each task (decision tree, random forests, xgboost, and choose another one yourself). The decision tree, random forests, and xgboost algorithms in the scikit-learn package can handle regression and classification tasks.

in the jupyter notebook(run_ml.ipynb)

2.2: Train your algorithms with the train and validation sets using 10-fold cross-validation (Figure 1). Note that you should choose appropriate metrics for comparing performance for each task. Name 3 metrics you could choose for LOS prediction. Name 3 metrics you could choose for Outcome prediction.

The three metrics I would use for LOS prediction are mean-squared error, explained variance score, and mean absolute error. The three metrics I would use for Outcome prediction are accuracy score, (precision, recall, and f1 score) and confusion matrix.

2.3 Classification/regression algorithms can have several hyperparameters, and finding the optimal combination of hyperparameters can significantly improve the model performance. For each algorithm for each task, come up

with two combinations of hyperparameters and provide your classification/regression results (accuracy/mean square error) here. Since you are doing cross-validation, your score results should be in the form of (mean±standard deviation). Your answer should be formatted as follows:
in the jupyter notebook(run_ml.ipynb)

```
LOS Prediction:
1. Decision Tree: (random_state: 42, max_depth: 5),  52.2216 ± 20.9596
                  (random_state: 42, max_depth: 10),  62.3406 ± 21.6673
2. Random Forest: (n_estimators: 50, random_state: 42, max_depth: 5),  40.3391 ± 17.6600
                  (n_estimators: 50, random_state: 42, max_depth: 10),  40.1438 ± 17.3493
3. XgBoost: (booster: gbtree, random_state: 42, max_depth: 5),  42.2067 ± 13.7943
                  (booster: dart, random_state: 42, max_depth: 10),  44.0573 ± 18.4344
4. SVM: (kernel: rbf, C: 0.1, gamma: scale),  43.9145 ± 19.2166
                  (kernel: rbf, C: 1.0, gamma: auto),  48.5667 ± 17.6198


Outcome  Prediction:
1. Decision Tree: (random_state: 42, max_depth: 5),  0.9094 ± 0.0662
                  (random_state: 42, max_depth: 10),  0.9004 ± 0.0801
2. Random Forest: (n_estimators: 50, random_state: 42, max_depth: 5),  0.9640 ± 0.0345
                  (n_estimators: 50, random_state: 42, max_depth: 10),  0.9670 ± 0.0409
3. XgBoost: (booster: gbtree, random_state: 42, max_depth: 5),  0.9730 ± 0.0365
                  (booster: dart, random_state: 42, max_depth: 10),  0.9730 ± 0.0365
4. SVM: (kernel: rbf, C: 0.1, gamma: scale),  0.8944 ± 0.0650
                  (kernel: rbf, C: 1.0, gamma: auto),  0.5438 ± 0.0517
```

2.4 Based on your answer to Question 2.3, which algorithm has the best performance for each task? Choose the best algorithm for the LOS prediction task and the best algorithm for the Outcome prediction task. Test them with the unseen test dataset provided ("datasets/test.csv") for each task, and provide your regression/classification results here based on 3 performance metrics.

in the jupyter notebook(run_ml.ipynb)
Random forest has the best performance for the regression task and XgBoost has the best performance for the classification task.

```
LOS Predictions:
Random Forest: Mean-squared error score, 28.33211687765665
                Explained variance score, 0.16509921147500317
                Mean Absolute Error, 3.8037560532800327
```

```
Outcome Predictions:
XgBoost: Accuracy Score: 1.0
         (Precision, recall, f1): (1.0, 1.0, 1.0)
          Confusion Matrix: [array([15,  0]), array([ 0, 15])]
```