INFO-I526 APPLIED MACHINE LEARNING – SPRING 2017
Carlos Sathler (cssathler@gmail.com)

Programming Assignment #2

1. Use adult.data as your training set and adult.test as your test set.
   We used the following data for this project:
   Training dataset: adult.data
   Test dataset: adult.test
   Both files are available from http://archive.ics.uci.edu/ml/datasets/Adult

2. You may want to consider creating a validation set from the training set to help you select hyper-parameters. Hyper-parameters you may want to tune are:
   a) the value of C in SVM;
   b) the depth of the trees in Adaboost and Random Forest;
   c) the number of trees in Adaboost and Random Forest.

   The following pseudo-code describe the tuning and training process we used for each algorithm:
   Pre-process Train and Test dataset in 5 different ways (see section 4.b), creating 5 versions of datasets.
   For each dataset version:
   Create train' and test' partitions by splitting the Train dataset (80/20% split)
   Use 10 fold validation to tune algorithm hyper parameters using train' partition of Train dataset
   Calculate mean f1-score from 10 fold validation exercise for each hyper parameter setting
   Use best hyper parameter setting to retrain the model on the entire train' partition
   Test model on test' partition and calculate f1-score for dataset version
   Capture dataset version yielding the best score

   Retrain model with best hyper parameter setting on the best performing Train dataset version
   Print scores
   Test model on the corresponding version of the Test dataset
   Print scores

   Note:
   - Please refer to Appendix A for code output.
   - Code was uploaded to Canvas along with PA#2 assignment report.

   The following table lists hyper parameters tuned for each algorithm:

Algorithm	Parameter
SVM	C
Adaboost	Max tree depth and number of trees
Random Forest	Max tree depth and number of trees

3. Take a look at the data and get a feel for the types of feature values you observe.
   We include in Appendix B a series of plots generated from Weka from a load of test.data in csv format into Weka. Please refer to Appendix B.

4. Preprocess the data. Make sure to explain how you preprocessed the data and why.

   a) It is important to note that some of the features are missing values (indicated with a "?"). How will you handle missing feature values? Explain your solution and why you chose it. You can treat a missing feature value as its own feature value, but this might not be the best solution, so you should look into others.

We identified missing values (column value = "?") in the following columns: workclass, occupation and native_country. These are all categorical features. Even though we could have used mode as a measure of centrality to impute missing values we opted to drop rows containing missing data to avoid adding bias to the training dataset.

b) Some of the feature values are continuous and some are categorical. Do you want to discretize the continuous features? If so, why? Do you want to discretize the feature values for all of the classifiers? It is okay to preprocess the data differently for each of the classifiers, just make sure to explain what you did and why.

In PA#1 we noticed that some of the algorithms outperformed when fed data that apparently should yield poorer results. For example, the tree classifier in PA#1 performed better when we used continuous variables, while we were under the impression from our class it would perform better if we used discretized features (since calculation of entropy would require discrete features).

The sklearn man pages do not make any recommendations about how to pre-process features for each algorithm, and from our research it doesn't appear there is an easy answer, or cookie-cutter approach on how to pre-process features for each algorithm. Apparently a lot will happen "under the hood" depending on how each sklearn algorithm is implemented.

With that in mind, we chose an empirical approach for PA#2; we decided to pre-process the data in 5 different ways, and we tested each algorithm on each dataset version. The table below describes how the data was pre-processed. Each algorithm was evaluated on each version of the dataset, each version corresponding to a different pre-processing approach.

Dataset Version	Continuous Feature	Ordinal Features	Categorical Features
Pre-processing 0	Leave unchanged	Leave unchanged	Leave unchanged
Pre-processing 1	Discretize	Leave unchanged	Leave unchanged
Pre-processing 2	Standardize	Standardize	Leave unchanged
Pre-processing 3	Discretize	Leave unchanged	Create dummy vars
Pre-processing 4	Standardize	Standardize	Create dummy vars

Feature	Type	General preprocessing	Pre-Pro 0	Pre-Pro 1	Pre-Pro 2	Pre-Pro 3	Pre-Pro 4
age	Numeric (Continuous)	Convert to float	No change	Discretize	Standardize	Discretize	Standardize
workclass	Categorical	Convert to float	No change	No change	No change	Dummy vars	Dummy vars
fnlwgt	Numeric (Continuous)	Convert to float	No change	Discretize	Standardize	Discretize	Standardize
education	Categorical	Remove (redundant)	N/A	N/A	N/A	N/A	N/A
educational-num	Numeric (Ordinal)	Convert to float	No change	No change	Standardize	No change	Standardize
marital-status	Categorical	Convert to float	No change	No change	No change	Dummy vars	Dummy vars
occupation	Categorical	Convert to float	No change	No change	No change	Dummy vars	Dummy vars
relationship	Categorical	Convert to float	No change	No change	No change	Dummy vars	Dummy vars
race	Categorical	Convert to float	No change	No change	No change	Dummy vars	Dummy vars
gender	Categorical	Convert to float	No change	No change	No change	Dummy vars	Dummy vars
capital-gain	Numeric (Continuous)	Convert to float	No change	Discretize	Standardize	Discretize	Standardize
capital-loss	Numeric (Continuous)	Convert to float	No change	Discretize	Standardize	Discretize	Standardize
hours-per-week	Numeric (Continuous)	Convert to float	No change	Discretize	Standardize	Discretize	Standardize

Feature	Type	General preprocessing	Pre-Pro 0	Pre-Pro 1	Pre-Pro 2	Pre-Pro 3	Pre-Pro 4
native-country	Categorical	Convert to float	No change	No change	No change	Dummy vars	Dummy vars
income	Binary flag	Convert to float	N/A	N/A	N/A	N/A	N/A

5. If you are using Weka, then convert the CSV into the appropriate ARFF format. If you are using Python, then I recommend looking into Numpy's loadtxt() function.
   I only used Weka to plot features in the train data set (see Appendix B).

6. If you are using Python/SKlearn, then implement the learning algorithm.
   Learning algorithms were implemented with Python and sklearn.
   Code was uploaded to Canvas along with PA#2 assignment report (file name = adult_pa2.py).

7. Learn the model using the training set and tune hyper-parameters on the validation set.
   Please refer to code uploaded to Canvas with assignment. See, for example, function "learn_svm".

8. Evaluate the learned model on the test set.

   a) If you used a validation set to tune your hyper-parameters, then make sure to use the model with the best performance on the test set.
   Each model is tuned with 10 kfold validation and tested on the train dataset validation partition (20% of training data) as explained under the answer for Question 2. Tuning parameters are also described under Question 2 above. Program output is included in Appendix A.
   Note that instead of running only the best model on the test dataset (file "adult.test") we ran all models against it, always using the best hyper parameter setting identified during training and tuning. This was done for illustrative purposes.
   As explained in the answer to Question 2, prior to running the learned model against the test dataset we learn the model against the entire training set with the proper hyper parameter setting.
   Also, as explained in the answers to questions 2 and 4 we tune the model against 5 versions of the data, each version defined by how the data was pre-processed. We evaluated not only parameter setting for each algorithm, but also what pre-processing yields the best results for each algorithm.
   Training results are listed in the program output in Appendix A under the title "TRAIN DATASET RESULTS".
   Test results are listed in the program output in Appendix A under the title "TEST DATASET RESULTS".
   Results identify best dataset version (best pre-processing) for each algorithm, along with best hyper parameter values for the algorithm/dataset.

9. Report the performance of the model.

   a) Make sure to use an appropriate performance measure (ie. accuracy vs. ROC vs. F1-score). Explain why you chose that performance measure.
   Since the proportion of adults with salary >$50k vs. salary <=$50 is slightly unbalanced (approximately 29% proportion with salary >$50k) we chose to use F1-score rather than accuracy to evaluate performance. We tuned all algorithms to capture the hyper parameter yielding the best F1-score.

   b) Explain the effect different hyper-parameters have on the performance of the algorithm. For example, what effect does the depth of the tree have on the performance of the decision tree?

Algorithm	Parameter	Effect
SVM	C	Penalty parameter of error term. Lower C reduces bias and increase variance.
Adaboost	Max tree depth and number of trees ("individual classifiers")	Low tree depth increase risk of underfitting and increase bias for individual classifiers, but boosting will reduce that tendency through weighted voting based on individual classifiers' accuracy.
Random Forest	Max tree depth and number of trees ("individual classifiers")	High tree depth increase risk of overfitting and increase variance for individual classifiers, but bagging will counter that tendency through simple majority voting which will cause random errors to cancel each other while reinforcing correct decision.

10. Briefly compare the performance of the different learning algorithms on the test set and make some hypothesizes about why you might have observed these differences.

The table below summarizes our results. We attribute performance differences among the classifiers to undetermined characteristics of the dataset features and their relationship to each other and the target label.
Best performance in each metric in the training set is highlighted in light blue.
Best performance in each metric in the test set is highlighted in light green.
Bold font indicates an improvement or no performance change for the metric on the test dataset (vs. train dataset).

Metric	SVM Pre-Processing = 4 C = 1		Adaboost Pre-Processing = 4 Trees = 10 Max Depth = 6		Random Forest Pre-Processing = 4 Trees = 50 Max Depth = 12	
	Train	Test	Train	Test	Train	Test
Accuracy	0.851	**0.853**	0.876	0.848	0.870	0.856
Error	0.149	0.147	0.124	0.152	0.130	0.144
Precision	0.759	0.758	0.787	0.725	0.830	0.791
Recall	0.591	0.588	0.689	0.612	0.602	0.562
F1-Score	0.665	0.662	0.735	0.664	0.698	0.657
ROC AUC	0.764	**0.764**	0.814	0.768	0.781	0.757

The Random Forest algorithm yielded the best accuracy and best precision scores while Adaboost yielded a lower accuracy and precision. SVM outperformed Adaboost on accuracy and precision but Adaboost produced the best recall, F1 and ROC AUC scores. In a production setting we would choose between Adaboost and Random Forest for this problem domain. Random Forest accuracy is less than 1% higher than Adaboost, but precision is 9.1% higher. On the other hand Adaboost recall is 8.9% higher with F1 Score and ROC AUC score also slightly higher. The final model would be chosen based on FP vs. FN cost evaluation.

It is interesting to note that, as expected, the high variance of deeper trees for the Random Forest algorithm (max depth = 12) is reduced by the high number of estimators (trees = 50), so that scores on test set don't show significant overfitting of the model in the end. The best Adaboost max tree depth was set at a much lower depth level of 6. The reasonably good results obtained in the end show that 10 estimators effectively countered bias and the model did not underfit.

All algorithms performed best with dataset version 4, i.e., standardized numeric features (continuous and encoded ordinal education number) and dummy variables for categorical features.

On a final note, we point out that our results are comparable to the results reported on the web site where the datasets were published (https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.names). The best result reported on the site shows accuracy=0.8595 for "FSS Naïve Bayes". That is less than 0.5% improvement over our best accuracy result with the SVM algorithm (C=1) run against standardized numerical features (continuous and ordinal) and dummy variables to represent categorical features (pre-processing option 4).

References:

[1] McKinney, Wes. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. " O'Reilly Media, Inc.", 2012.

[2] James, Gareth, et al. *An introduction to statistical learning*. Vol. 6. New York: springer, 2013.

[3] Raschka, Sebastian. *Python machine learning*. Packt Publishing Ltd, 2015.

[4] Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. Springer, Berlin: Springer series in statistics, 2001.

[5] Downey, Allen. *Think Python*. " O'Reilly Media, Inc.", 2012.

APPENDIX A

```
RESULTS FOR SVM CLASSIFIER
--------------------------
Best pre-processing option: 4
Best C = 1

           TRAIN DATASET RESULTS
           Accuracy........: 0.851
           Precision Score.: 0.759
           Recall Score....: 0.591
           F1 Score........: 0.665
           ROC AUC.........: 0.764

           Confusion Matrix:
           +----------------------------+
           |         Prediction         |
           +----------------------------+
           |      0      |       1      |
      +---|----------------------------|
      | 0 |    21243    |      1411    |
Truth +   |----------------------------|
      | 1 |     3070    |      4438    |
      +---+----------------------------+

           TEST DATASET RESULTS
           Accuracy........: 0.853
           Precision Score.: 0.758
           Recall Score....: 0.588
           F1 Score........: 0.662
           ROC AUC.........: 0.764

           Confusion Matrix:
           +----------------------------+
           |         Prediction         |
           +----------------------------+
           |      0      |       1      |
      +---|----------------------------|
      | 0 |    10666    |       694    |
Truth +   |----------------------------|
      | 1 |     1524    |      2176    |
      +---+----------------------------+

RESULTS FOR ADABOOST CLASSIFIER
-------------------------------
Best pre-processing option: 4
Best no of estimators = 10
Best max tree depth = 6

           TRAIN DATASET RESULTS
           Accuracy........: 0.876
           Precision Score.: 0.787
           Recall Score....: 0.689
           F1 Score........: 0.735
           ROC AUC.........: 0.814

           Confusion Matrix:
           +----------------------------+
           |         Prediction         |
           +----------------------------+
           |      0      |       1      |
      +---|----------------------------|
      | 0 |    21254    |      1400    |
Truth +   |----------------------------|
      | 1 |     2335    |      5173    |
```

```
       +---+----------------------------+

           TEST DATASET RESULTS
           Accuracy........: 0.848
           Precision Score.: 0.725
           Recall Score....: 0.612
           F1 Score........: 0.664
           ROC AUC.........: 0.768

           Confusion Matrix:
           +----------------------------+
           |         Prediction         |
           +----------------------------+
           |      0      |       1      |
       +---|----------------------------|
       | 0 |    10503    |       857    |
 Truth +   |----------------------------|
       | 1 |     1437    |      2263    |
       +---+----------------------------+

RESULTS FOR RANDOM FOREST CLASSIFIER
------------------------------------
Best pre-processing option: 4
Best no of estimators = 50
Best max tree depth = 12

           TRAIN DATASET RESULTS
           Accuracy........: 0.870
           Precision Score.: 0.830
           Recall Score....: 0.602
           F1 Score........: 0.698
           ROC AUC.........: 0.781

           Confusion Matrix:
           +----------------------------+
           |         Prediction         |
           +----------------------------+
           |      0      |       1      |
       +---|----------------------------|
       | 0 |    21730    |       924    |
 Truth +   |----------------------------|
       | 1 |     2986    |      4522    |
       +---+----------------------------+

           TEST DATASET RESULTS
           Accuracy........: 0.856
           Precision Score.: 0.791
           Recall Score....: 0.562
           F1 Score........: 0.657
           ROC AUC.........: 0.757

           Confusion Matrix:
           +----------------------------+
           |         Prediction         |
           +----------------------------+
           |      0      |       1      |
       +---|----------------------------|
       | 0 |    10811    |       549    |
 Truth +   |----------------------------|
       | 1 |     1620    |      2080    |
       +---+----------------------------+
```

INFO-I526 APPLIED MACHINE LEARNING – SPRING 2017

Carlos Sathler (cssathler@gmail.com)

APPENDIX B