**Homework 2**

Due Oct 25, 2022

Wine-Tasting Machine

In this assignment, we will practice building supervised machine learning with Logistic Regression (LR), Naïve Bayes (NB), Support Vector Machine (SVM), and Decision tree (DT), Random Forest (RF) classifiers, as compared with simple/baseline methods such as OneR and ZeroR. The data for this exercise comes from the wine industry. Each record represents a sample of a specific wine product, the input attributes include its organoleptic characteristics, and the output denotes the quality class of each wine: {high, low}. The labels have been assigned by human wine-tasting experts, and we can treat that information as “ground truth” in this exercise. Your job is to build the best model to predict wine quality from its characteristics, so that the winery could replace the costly services of professional sommeliers with your automated alternative, to enable quick and effective quality tracking of their wines at production facilities. They need to know whether such change is feasible, and what extent of inaccuracies may be involved in using your tool.

You will be asked to run experiments in both WEKA and Python.

You are given two datasets red-wine.csv and white-wine.csv: [Dataset folder](https://drive.google.com/drive/folders/1IndS9vELiONWwamkmQxvjkeLJtSSFXa1?usp=sharing)

Deliverable:

* A copy of word doc with answers (including screenshots) to blackboard
* Python Notebook to be uploaded to GitHub and shared with instructor/TA, as with homework 1

**WEKA Tasks (50 points)**

1. **Load red-wine.csv into WEKA (15 points)**
   1. **Create a conditional distribution for each of the input variables with respect to output (click the “visuali zing all” button, making sure you set the output correctly)**

Chart, histogram

Description automatically generated

* 1. **Comparing the plots for Sulphates and Alcohol, which one do you think is more predictive of the wine quality, and why?**

These are the plots for sulphates and alcohol and when Comparing the plots for sulphate and Alcohol, Sulphate is more predictive of the wine quality

Chart, histogram

Description automatically generatedChart, bar chart

Description automatically generated

* 1. **Verify your answer by using a logistic regression model, is it consistent with your speculation in (b)? (hint: here you may use univariate logistic regression, the better performance, the more predictive a feature is. AUC is a good score for this purpose)**

Correctly Classified Instances i.e. Accuracy for Logistic Regression for Alcohol is 51.4886%

AUC Score for Alcohol is 0.436

Text

Description automatically generated

Correctly Classified Instances i.e. Accuracy for Logistic Regression for Sulphate is 75.7583%

AUC Score for Sulphate is 0.867

Text

Description automatically generated

We have the Better performance and high Accuracy for Sulpate i.e.77.7583% and the good AUC Score for sulphate i.e.., 0.867.

By using Logistic regression model sulphate has better performance,highly predictive feature and good AUC score.

1. **Fit a model using each of the following methods and report the performance metrics of 10-fold cross-validation using red-wine.csv as the training set (25 points)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Model | ZeroR | OneR | LR | NB | DT | SVM | RF |
| AUC | 0.496 | 0.785 | 0.873 | 0.890 | 0.877 | 0.784 | 0.920 |
| Accuracy | 52.8897 % | 78.4588 % | 79.3345 % | 82.4869% | 84.238% | 78.634 % | 85.289 % |

1. **Obtain the ROC curve for the best-performing model in terms of AUC score from the experiment above, paste a screenshot here and comment on its performance (5 points)**

After reporting the performance metrics we got the high accuracy and good AUC Score for random forest, so the best perfoming model is Random Forest.

Graphical user interface, application, Word

Description automatically generated

1. **Using the best model obtained above in WEKA and run the model on white-wine.csv and report the AUC score, comment on the performance. (hint: see WEKA reference section on how to get performance on an external independent test set) ( 5 points)**

The best model obtained in WEKA is Random Forest. After running the white-wine.csv we got the Accuracy of 87.1795% and AUC Score is 0.956.

Text

Description automatically generated

**Python Tasks (50 points)**

Submission: Upload Python Notebook to GitHub links as homework 1

Github Link:- https://github.com/ThanmayeePola/Data\_Mining

1. Read **red-wine.csv** into Python as a data frame, use a pandas profiling tool (<https://github.com/pandas-profiling/pandas-profiling>) to create an HTML file, and paste a screenshot of the HTML file here (10 points)
2. Repeat the same experiments in WEKA Question 2, and report the same metrics as in Question 2. To receive full credit, you will need to write a script to assemble the result as above in the form of Pandas data frame. Paste a screenshot of your result from your Python notebook here. Please make sure that there is a reasonable number of significant digits in reporting your output. (20 points)
3. Plot the ROC curve of the Random Forest classifier from the Python package, and paste a screenshot of your ROC curve here (10 points)
4. Using the best model obtained above in Q2 (python) and running the model on **white-wine.csv** and reporting the AUC score, comment on the performance. (5 points)
5. Suppose all the models have comparable performance, which model would you prefer if the wine-tasting experts would like to gain some insights into the model? Note: there could be multiple model types fitting this criterion. (5 points)

Reference (WEKA)

**How to find Classifiers**

* Logistic regression: function -> logistic
* SVM: function -> libSVM or SMO
* Naive Bayes: Bayes -> Naive Bayes
* Decision Trees: Trees -> J48
* Random Forest: Trees -> Random Forest

**Obtain performance on an external test set**

**1.** Run WEKA and select the Explorer application.

**2.** Under the Preprocess tab, select "Open file..." and open the ‘red\_wine.csv’ dataset. You should now see information about this dataset.

**3.** Under the "Classify" tab, click on "Choose" and select the classifier of your choice

**4.** Select ‘Test options’ -> supplied test set, load ‘white\_wine.csv’ dataset.

**Experimenter**

Following steps are provided in case you want to use Experimenter.

Use this option if you have a large number of experiments and a large number of datasets.

Perform the following experiment comparing some classifiers using default parameter settings on the datasets: contact-lenses.arff, segment-challenge.arff, soybean.arff, weather.arff, and vote.arff.

a. In WEKA, choose the "Experimenter" application.

b. Under "Experiment Configuration Mode" choose "New".

c. Choose the default Experiment Type: 10-fold cross-validation and classification.

d. Choose default Iteration Control: 10 repetitions and data sets first.

e. Under "Experiment Type" choose "Train/Test Percentage Split (data randomized)".

f. Under "Datasets", choose "Add new..." for each of the 4 datasets. Note that the segment-test dataset has been removed from the list.

g. Under "Algorithms", choose "Add new..." and add ‘J48’ (under trees), IBk (under lazy), NaiveBayes (under bayes), SMO (under functions), and Logistic (under functions) one after another. Use default parameters for each except IBk (select k=3 for IBk).

h. Now click on "Run" at the top, and then click "Start".

i. Now click on "Analyse" at the top.

j. Under "Source", choose "Experiment".

k. Near the bottom, click on "Perform test". Analyze the results using the following configuration

* Testing with: Paired T-Tester (corrected)
* Comparison field: Percent\_correct
* Significance: 0.05
* Test base: bayes.NaiveBayes
* Show std. deviations: (checked)

Reference (Python)

* Run K-fold cross-validation experiment
  + <https://www.askpython.com/python/examples/k-fold-cross-validation>
* Fitting model and compute AUC/ROC

<https://www.youtube.com/watch?v=uVJXPPrWRJ0>

* Baseline model OneR and ZeroR - you may refer to https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html

Or you may implement your own version

* Model fitting
  + You should be able to find all the following model fitting functions from the sklearn package

<https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html>

<https://scikit-learn.org/stable/modules/naive_bayes.html>

<https://scikit-learn.org/stable/modules/svm.html#svm-classification>

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

<https://scikit-learn.org/stable/modules/tree.html>