Paper Critique

CP610 Data Analysis Spring 2021

1. Introduction
2. Summary
3. Critique
4. Conclusion

Dee Wu
Physics and Computer Science (MCS)
Wilfrid Laurier University
Waterloo, Canada
wuxx2586@mylaurier.ca

1. Introduction

- "*Data Mining Approach to Detect Heart Diseases*", IJACSIT
  - Vikas Chaurasia, Sai Nath University
  - Saurabh Pal, VBS Purvanchal University
- UCI Heart Disease Data Set
- Algorithm: Naïve Bayes, J48 Decision Tree, and Bagging
- Third Party Tool: Weka and Kappa
- Limitations and the validation of data

2. Summary

- Analyzed different classifiers in the diagnosis of heart diseases
- Evaluated the performance by the confusion matrixes
- The comparisons show the bagging algorithm is the winner
- The time consumed to build the model is slightly longer

3. Critique - Strengths

- The classification models are suitable
  - Naïve Bayes, J48
  - Decision Tree, and
  - Bagging
- The third-party tools.
  - Kappa statistic - evaluate the accuracy of the measuring cases
  - Weka - implement the three nominated classifiers

3. Critique - Weakness

- Applying the advanced methods
- The directions for the left unfinished work
- The credibility of the data set

The file cleveland.data has been unfortunately messed up when we lost node cip2 and loaded the file on node ics. The file processed.cleveland.data seems to be in good shape and is useable (for the 14 attributes situation). I'll clean up cleveland.data as soon as possible.

Bad news: my original copy of the database appears to be corrupted. I'll have to go back to the donor to get a new copy.

David Aha

4. Conclusion

- The research work achieves the objective, that to predict accurate presence of heart disease with reduced number of attributes, by applying Naïve Bayes, J48 Decision Tree, and Bagging algorithms and the 10-fold cross validation method.

- The areas for improvement could be adding more analysis and validation of the data set, as well as the directions for the left unfinished work in the medical data mining area.

Diagnosis of Heart Disease

CP610 Data Analysis Spring 2021

1. Problem definition and motivation
2. The approach
3. Performance and analysis
4. Experimental Result and Discussion
5. Conclusion

Dee Wu
Physics and Computer Science (MCS)
Wilfrid Laurier University
Waterloo, Canada
wuxx2586@mylaurier.ca

1. Problem definition and motivation

- Analyzed UCI Heart Disease Data Set
- Utilizing data analysis model/ algorithms
- Evaluation and comparison of different algorithms
- The optimizations in necessity
- Applied data visualization libraries
- Developed a Machine Learning API using Flask Microservice

2. The approach - DATA

- Andrew Ng, one of the forerunners in AI, tweeted

  "AI Systems = Code (model/algorithm) + Data”

- UCI Machine Learning Repository, Heart Disease Data Set
  - Cleveland Clinic Foundation (cleveland.data, processed.hungarian.data)
  - Hungarian Institute of Cardiology, Budapest (hungarian.data, processed.hungarian.data)
  - V.A. Medical Center, Long Beach, CA (long-beach-va.data, processed.va.data)
  - University Hospital, Zurich, Switzerland (switzerland.data, processed.switzerland.data)

- The databases have 76 raw attributes, only 14 of them are used

2. The approach - Algorithm

- Good accuracy algorithm
  - Random Forest Classifier
  - Decision Tree
  - Naive Bayes
- Poor accuracy algorithms
  - Support Vector Machine
  - Logistic Regression
  - KNN Classifier
- Ensemble Learning or Bagging algorithms performs the best

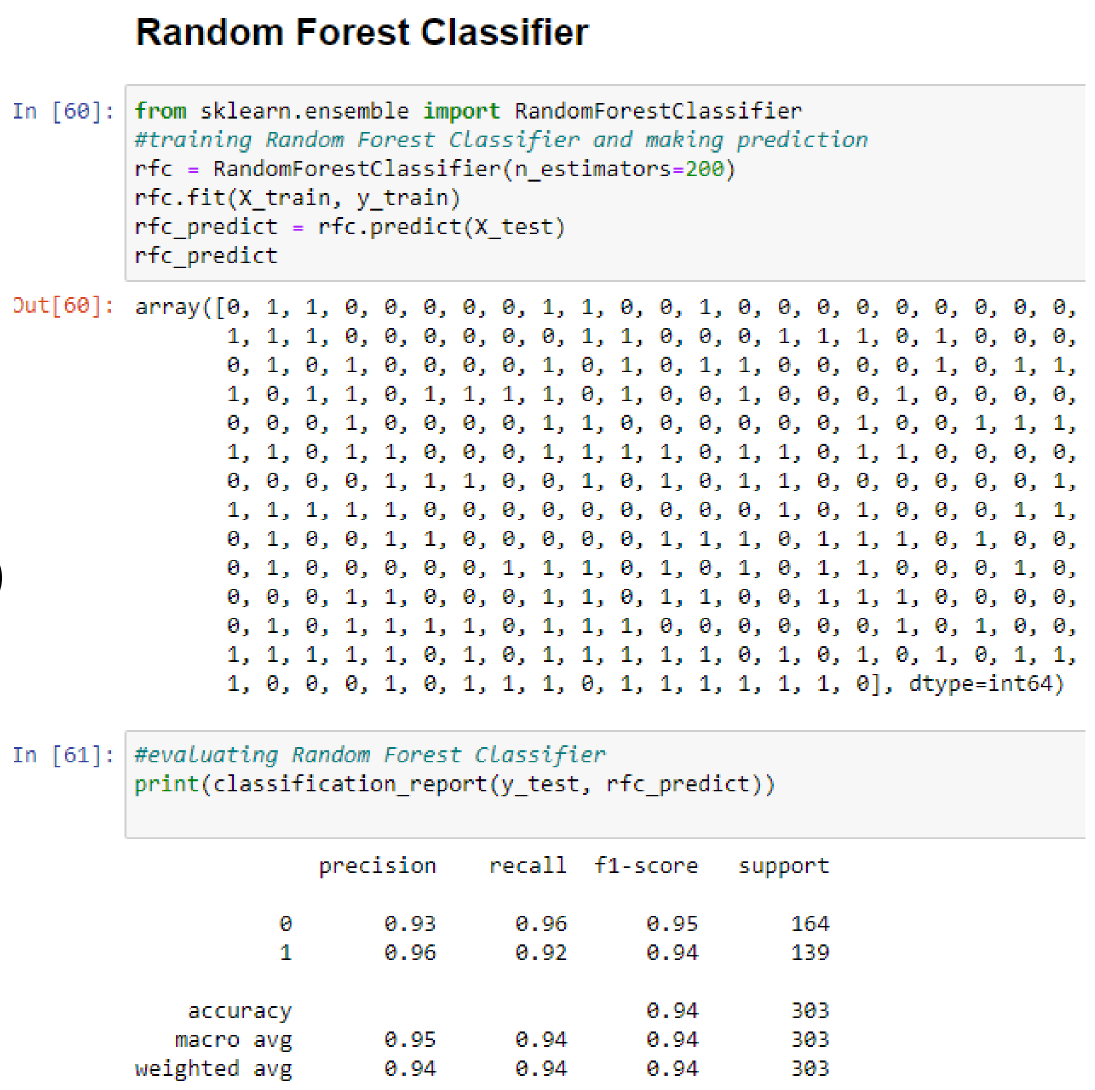
3. Performance and analysis

- Performance and analysis
  - Prediction
  - Confusion Matrix
- Algorithm Optimization
  - Hyperparameter Optimization
  - Feature Normalization and Standardization

0.8
88

3.1 Performance and analysis

- Random Forest Classifier (94%)
- Decision Tree (100%)
- Naive Bayes (100%)
- Support Vector Machine (51%)
- Logistic Regression (52%, 76%, 97%)
- KNN Classifier (52%)
- Ensemble Learning or Bagging
- (79.54%, 90.76%, 100%)

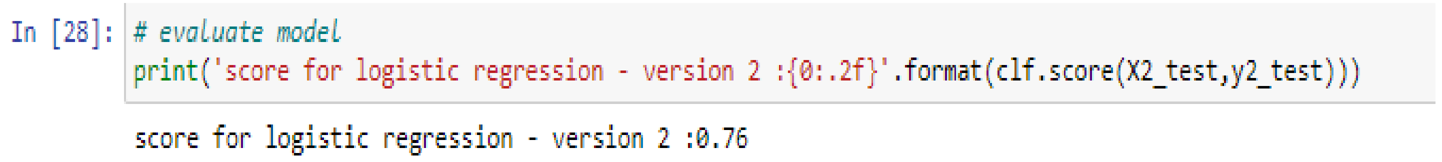


Random Forest Classifier

```
In [60]: from sklearn.ensemble import RandomForestClassifier
         #training Random Forest Classifier and making prediction
         rfc = RandomForestClassifier(n_estimators=200)
         rfc.fit(X_train, y_train)
         rfc_predict = rfc.predict(X_test)
         rfc_predict

Out[60]: array([0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0,
                0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1,
                1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
                0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1,
                1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0,
                0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1,
                1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1,
                0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0,
                0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0,
                0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0,
                0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
                1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1,
                1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0], dtype=int64)

In [61]: #evaluating Random Forest Classifier
         print(classification_report(y_test, rfc_predict))

              precision    recall  f1-score   support

           0       0.93      0.96      0.95       164
           1       0.96      0.92      0.94       139

    accuracy                           0.94       303
   macro avg       0.95      0.94      0.94       303
weighted avg       0.94      0.94      0.94       303
```

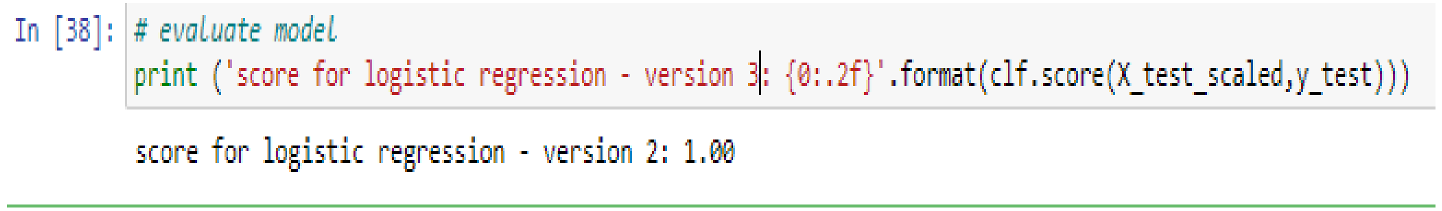
3.2 Algorithm Optimization

- Hyperparameter Optimization



```
In [28]: # evaluate model
         print('score for logistic regression - version 2 :{0:.2f}'.format(clf.score(X2_test,y2_test)))

         score for logistic regression - version 2 :0.76
```

- Feature Normalization and Standardization



```
In [38]: # evaluate model
         print ('score for logistic regression - version 3: {0:.2f}'.format(clf.score(X_test_scaled,y_test)))

         score for logistic regression - version 2: 1.00
```

4.Experimental Result and Discussion

- *Data Cleansing*
- *Model evaluation*
- *Visualization*
- Heart Disease Diagnosis API

4.1 Data Cleansing

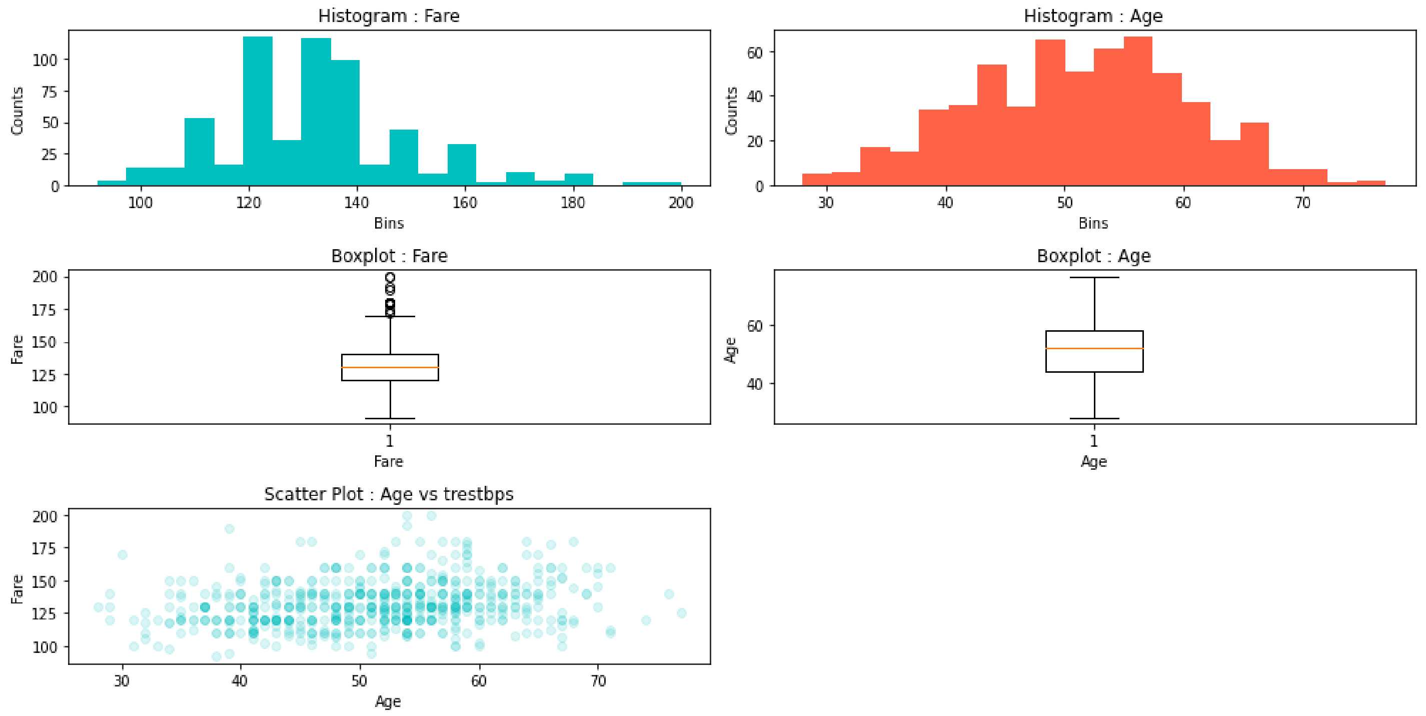
- Exploratory Data Analysis
- Data Munging
- Feature Engineering

4.2 Model evaluation

- Import sklearn and other Python libraries
- Implement confusion metrics
  - Accuracy, macro avg, weighted avg
  - Precision, recall, f1-score
- Describe generalization error in scope

4.3 Visualization

- Matplotlib
  - visualize the result of data mining
  - customize any aspects of the chart
- Complicated visualization
  - ax_arr vs. axes
  - two-dimensional array indexing



4.4 Heart Disease Diagnosis API

- In REST API
  - Client make HTTP request
  - Server can send back the HTTP response
- Create API, and invoke the API (Requests library)
  1. The API hosted on the server extract the input data from the Flask request object
  2. Persisted model is loaded to make predictions
  3. The predictions is returned from the model
  4. It will be sent back to the client wrapped in an HTTP response object

5. Conclusion

- Analyzed UCI Heart Disease Data Set to predict heart disease diagnosis
  - Utilize the data analysis model/ algorithms
  - Confusion matrix and the comparisons
  - Optimization methods
- Implemented by Jupyter Notebook IDE
  - Python with the imported libraries
- The Data Analysis/ Machine Learning model/ algorithms
  - Random Forest Classifier, Support Vector Machine, Logistic Regression, KNN Classifier, Naïve Bayes, Decision Tree, Ensemble Learning, Bagging algorithm
- Machine Learning API is developed
  - Flask Microservice
  - pickle persisted model

Question?