Diagnosis of Heart Disease

CP610, A Data Analysis Project - Spring 2021

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

***Abstract*— Diagnosis of Heart Disease Project and the work analyzed UCI Heart Disease Data Set to predict heart disease diagnosis, by utilizing data analysis methodologies, with the evaluation and comparison of different algorithms and the optimizations in necessity, which also applied current data visualization libraries in data visualization, as well as developed a Machine Learning API using Flask Microservice.**

Keywords*— *Random Forest Classifier, Support Vector Machine, Logistic Regression, KNN Classifier, Naïve Bayes, Decision Tree, Ensemble Learning, Bagging algorithm, Flask

I. Introduction

The goal is to implement the fundamental machine learning and data analysis methodologies including supervised and unsupervised learnings, in which Diagnosis of Heart Disease is a good candidate to apply the algorithms for clustering to predict the diagnostic result.

In the project the following algorithms have been applied, Dummy Classifier, Logistic Regression Model, K-NN Classifier, Gradient Boosting Classifier, Bagging Classifier, and Area Under the ROC Curve (AUG). There will be an evaluation session for the algorithms applied and visualizations with Matplotlib and other libraries. To access prediction result as a RESTful request dispatching the project also creates a Python API by using Flask microframework.

II. Background

A. Data Collection and Data Analysis

Andrew Ng, one of the forerunners in AI, tweeted, "AI Systems = Code (model/algorithm) + Data".

The importance of data is never over estimated. In this project the data source is also critical as the quality of data decides the quality of the predictions. One dataset is from Kaggle [5] and another one from UCI [2], with credibility and completeness, and was chosen as data of this project.

B. Machine Learning and the Algorithms

In Supervised learning the training data is composed of labels and new data is classified based on the training set. In Unsupervised learning the class labels of the training data are not known, and clustering establishes the classes or clusters in the data. Heart Disease Diagnosis dataset is a labelled dataset, and the prediction is a Supervised learning function.

The regression models (e.g., Logistic Regression) performed badly, even after optimization the accuracy is still under 0.80 and the expectations. With the same test dataset some algorithms show accuracy equals to 1, revealing a overfitting pattern, which could be caused by the smaller size of the training dataset even it is a complete dataset instead of a splitting partial of it.

III. Heart Disease Data

" In particular, the Cleveland database is the only one that has been used by ML researchers to this date. The "goal" field refers to the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 4. Experiments with the Cleveland database have concentrated on simply attempting to distinguish presence (values 1,2,3,4) from absence (value 0).

The names and social security numbers of the patients were recently removed from the database, replaced with dummy values."[2]

A. Authors and Affiliations

The authors of the databases have requested:

...that any publications resulting from the use of the data include the names of the principal investigator responsible for the data collection at each institution. They would be:

- *Hungarian Institute of Cardiology. Budapest: Andras Janosi, M.D.*
- *University Hospital, Zurich, Switzerland: William Steinbrunn, M.D.*
- *University Hospital, Basel, Switzerland: Matthias Pfisterer, M.D.*
- *V.A. Medical Center, Long Beach and Cleveland Clinic Foundation: Robert Detrano, M.D., Ph.D." [2]*

B. Figures and Tables

This Heart Disease Dataset (UCI Machine Learning Repository) contains 4 databases, all attributes are numeric-valued. The data was collected in heart disease diagnosis by the following organizations located geographically separated.

- 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)

Each database has the same instance format. While the databases have 76 raw attributes, only 14 of them are used (Table 1.). Each dataset has a "processed" version, which only contains 14 attributes, although missing values still exists. In practical analysis the missing values in the processed dataset are replaced with value -888.0 to be distinguished from regular data.

TABLE I. DATASET ATTRIBUTES

No.	Table Column Head	
	Attribute	***Description***
1	age	age in years
2	sex	sex (1 = male; 0 = female)
3	cp	chest pain type -- Value 1: typical angina -- Value 2: atypical angina -- Value 3: non-anginal pain -- Value 4: asymptomatic
4	trestbps	resting blood pressure (in mm Hg on admission to the hospital)
5	chol	serum cholestoral in mg/dl
6	fbs	(fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
7	restecg	resting electrocardiographic results -- Value 0: normal -- Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV) -- Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria
8	chalach	maximum heart rate achieved
9	exang	exercise induced angina (1 = yes; 0 = no)
10	oldpeak	ST depression induced by exercise relative to rest
11	slope	the slope of the peak exercise ST segment -- Value 1: upsloping -- Value 2: flat -- Value 3: downsloping
12	ca	number of major vessels (0-3) colored by flourosopy
13	thal	3 = normal; 6 = fixed defect; 7 = reversable defect
14	num	diagnosis of heart disease (angiographic disease status) -- Value 0: < 50% diameter narrowing -- Value 1: > 50% diameter narrowing (in any major vessel: attributes 59 through 68 are vessels)

IV. DATA MINING MODEL

As a medical diagnosis problem Heart Disease Diagnosis is a typical classification problem that makes prediction based on the categorical class labels in the training set and the values in same, chosen by attribute selection measure: Information Gain (ID3/ C4.5).

In predicting the test set some algorithms have good accuracy, Random Forest Classifier, Decision Tree, Naive Bayes; some reveals accuracy not of satisfactory, Support Vector Machine, Logistic Regression, KNN Classifier, close to 80% after optimization; Ensemble Learning or Bagging algorithms performs the best and obviously a final winner of the choice.

A. *Algorithms show High Accuracy*

1) *Random Forest Classifier*

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
```

2) *Decision Tree*

Decision Tree

```
In [64]: from sklearn.tree import DecisionTreeClassifier
         # training Decision Tree Classifier
         decision_tree = DecisionTreeClassifier()
         decision_tree.fit(X_train, y_train)
         dt_predict = decision_tree.predict(X_test)
         dt_predict

Out[64]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)

In [65]: print(classification_report(y_test, dt_predict))
              precision    recall  f1-score   support

           0       0.55      0.63      0.59       164
           1       0.48      0.40      0.43       139

    accuracy                           0.52       303
   macro avg       0.52      0.51      0.51       303
weighted avg       0.52      0.52      0.52       303
```

3) *Naive Bayes*

Naive Bayes

```
In [117]: from sklearn.naive_bayes import GaussianNB
          ## Naive Bayes
          nb = GaussianNB()
          nb.fit(X_train, y_train)
          nb_predict = nb.predict(X_test)
          nb_predict

Out[117]: array([0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
                 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0,
                 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 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, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 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, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1,
                 0, 1, 0, 0, 1, 1, 0, 0, 1, 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, 0, 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, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
                 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 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 [118]: print(classification_report(y_test, nb_predict))

              precision    recall  f1-score   support

           0       1.00      1.00      1.00       164
           1       1.00      1.00      1.00       139

    accuracy                           1.00       303
   macro avg       1.00      1.00      1.00       303
weighted avg       1.00      1.00      1.00       303
```

B. *Algorithms show Low Accuracy*

1) *Support Vector Machine*

Support Vector Machine

```
In [102]: from sklearn.svm import SVC
          # training Support Vector Machine and making prediction
          svmc = SVC()
          svmc.fit(X_train,y_train)
          svmc_predict = svmc.predict(X_test)
          svmc_predict

Out[102]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)

In [103]: print(classification_report(y_test, svmc_predict))

              precision    recall  f1-score   support

           0       0.55      0.47      0.51       164
           1       0.47      0.55      0.51       139

    accuracy                           0.51       303
   macro avg       0.51      0.51      0.51       303
weighted avg       0.52      0.51      0.51       303
```

2) *Logistic Regression*

a) *Logistic Regression Model*

```
In [18]: # evaluate model
         print('score for logistic regression - version 1 :{0:.2f}'.format(model_lr_1.score(X_test, y_test)))

         score for logistic regression - version 1 :0.52
```

b) *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
```

c) *Feature Normalization and Standardization:*

```
In [50]: # evaluate model
         print ('score for logistic regression - version 3: {0:.2f}'.format(clf.score(X_test_scaled,y_test)))

         score for logistic regression - version 3: 0.97
```

3) *KNN Classifier*

KNN Classifier

```
In [110]: from sklearn.neighbors import KNeighborsClassifier
          neigh = KNeighborsClassifier(n_neighbors=12)
          result = neigh.fit(X_train, y_train)
          knn_predict = neigh.predict(X_test)
          knn_predict

Out[110]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0,
                 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)

In [111]: print(classification_report(y_test, knn_predict))

              precision    recall  f1-score   support

           0       0.55      0.59      0.57       164
           1       0.48      0.44      0.46       139

    accuracy                           0.52       303
   macro avg       0.52      0.52      0.51       303
weighted avg       0.52      0.52      0.52       303
```

C. *Ensemble Learning (Bagging)*

Bagging is implemented as one of the Ensemble Methods (the other two are Boosting, Random Forest), which uses a combination of models to create an improved model to improve accuracy. The voting classifier is composed of different models, or estimators, resembling the diagnosis based on multiple doctors' majority vote.

voting classifier that combines four different estimators

```
In [76]: from sklearn.ensemble import RandomForestClassifier, VotingClassifier
         evc = VotingClassifier(estimators=[('svm',svm),('nb',nb),('rf',rf),('lr',lr)],
                               voting='hard')
         evc.fit(X_train,y_train)
         evc.score(X_test, y_test)

Out[76]: 0.7953795379537953
```

In training a set D of d tuples is given, at each iteration i, a training set D_i of d tuples is sampled with replacement from D, and a classifier model M_i is learned for each training set D_i. In classification Each classifier Mi returns its class prediction, and the bagged classifier M* counts the votes and assigns the class with the most votes to X. [3]

bagging classifier with 100 random forest estimators

```
In [77]: from sklearn.ensemble import BaggingClassifier

         bg = BaggingClassifier(rf, max_samples=0.6, max_features=1.0, n_estimators=100)

         bg.fit(X_train, y_train)

         bg.predict(X_test)

         bg.score(X_test, y_test)
```

D. *Algorithm Optimization*

1) *Hyperparameter Optimization*

The scikit-learn function GridSearchCV is used to perform hyperparameter optimization. Firstly, a basic logistic regression model is created, then import GridSearchCV function. Next, a parameter dictionary is created where for each of the hyperparameter the values are listed out to try out during the optimization process. Like for C, a few values ranging from 1 to 1000 have been specified, and for the penalty parameter L1 and L2 are the tries. Once the

parameters set is obtained, it is time to create the grid search object, clf, using the GridSearchCV function.

In this function, the base model is specified, then the grid parameters are set using the param_grid attribute. The cv value is set to 3, and this will perform the 3-fold cross-validation. In this way the best_params property is used to get the best settings. In the code it is showing 10.0 for C and l2 for the penalty. The best_score_ property is resulted as 0.71 in this case.

2) Feature Normalization and Standardization

Many machine learning algorithms can work better if you provide features on the same scale. For a logistic regression model that we are building in this course, it may not make a significant impact, but if you are using more advanced machine learning algorithms, such as neural networks, then it is always suggested to perform feature normalization before fitting the data to the model.

Many of times, we also utilize feature standardization, especially when the machine learning algorithm not only bothered about the range, but also about the distribution. Suppose the distribution of each of these features are very different with different mean values and standard deviations. Then what we need to do is to standardize the features in such a way that all the features have 0. 0 mean and unit variance. Such a standardization can boost model performance, especially for those models that rely heavily on data distributions.

An excellent prediction results is seen in the example (Diagnosis of Heart Disease - 02 Model).

V. Experimental Result and Discussion

A. Data Cleansing

Data cleaning and organizing data occupy 60% data scientists spend their time, according to Crowdflower data science report 2016. The preprocessing tasks include Exploratory Data Analysis, Data Munging, Feature Engineering, and Advanced Visualization.

1) Exploratory Data Analysis

Exploratory Data Analysis (or ETA) is composed of five modules, Basic Structure, Summary Statistics, Distributions, Grouping, and Crosstabs, Pivots.

In practice the Basic Structure of heart disease dataset has been investigated, such as selection, indexing, and filtering. The centrality measurements have been calculated, such as Mean, Median, same as spread measurement, Range, Percentiles and Boxplot, Variance and Standard Deviation. Work has been done getting summary statistics for numerical features (counts and proportions) and summary statistics for categorical features.

Univariate distribution plots, e.g., Histogram and KDE Plot, bivariate distribution plot, e.g., Scatter Plot, Grouping, Aggregation, Crosstab, Pivot Table are fully investigated in practical work.

2) Data Munging

One of the problems in the dataset of Heart Disease Diagnosis is it needs to deal with the missing values, such as the issues and solution, imputation techniques. In practice the solution utilizes Python libraries, such as Pandas, Numpy, and the coverage even extended to detecting and treating outlier data.

3) Feature Engineering

The scope covers feature creation using Pandas and NumPy; categorical feature encoding: Binary Encoding, Label Encoding, One-hot Encoding; drop and reorder columns using Pandas; save Dataframe to file using Pandas, and reproducible script for data processing using Pandas and NumPy.

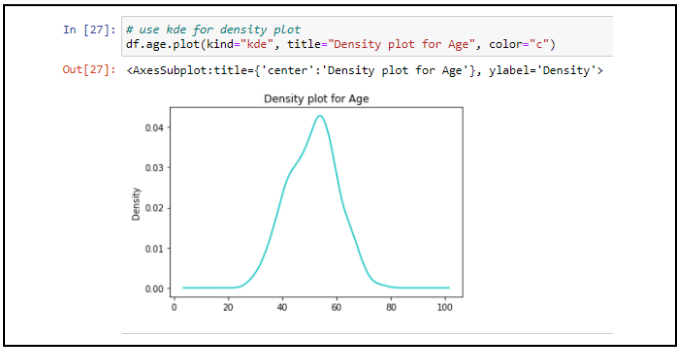
B. Model evaluation

Import sklearn and other Python libraries to evaluate the models in comparison.

- Implemented the model evaluation metrics, such as accuracy, precision.
- Described generalization errors in scope for the learning algorithms.

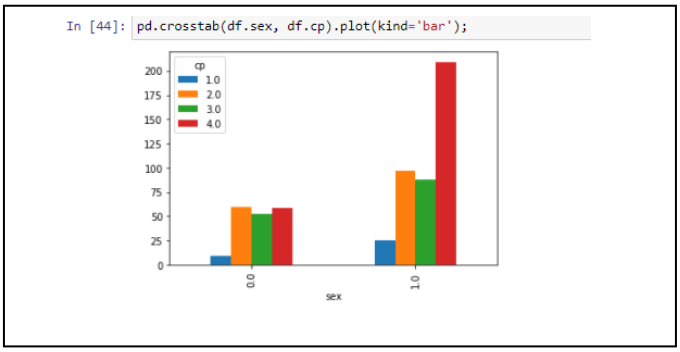
C. Visualization

Matplotlib is an excellent tool to visualize the result of data mining with the ability to customize any aspects of the chart. On more complicated visualization ax_arr is used instead of individual axes and accessed using a two-dimensional array indexing. In the example several plots are created here, two histograms, one for Fare, one for Age, then we are creating a couple of box plots, and finally, one scatter plot.



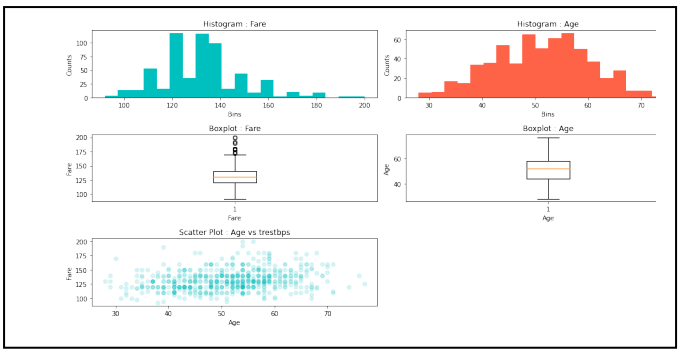
a. Density plot for Age.

Fig. 1. Use kind= "kde" for density plot.



b. Bar chart for Sex and Pclass.

Fig. 2. Crosstab on Sex and Pclass.



c. Adding subplots.

Fig. 3. Advanced Visualization.

In order to remove a few overlaps between the subplots, the plt. tight_layout function is used, which will add the subplots with some paddings.

D. Heart Disease Diagnosis API

In REST API, the client can make a request over the HTTP protocol, and the server can send back the HTTP response. Also, in the REST world, you can use the common HTTP verbs such as GET to get some information and POST to send some data and get the response.

In the data extraction module, usually an existing API is chosen, and invoked by the Python Requests library to get the data. However, in the practice we will create our own API, and then we will use the Requests library to invoke the API. The job of a machine learning API is to return model predictions when the input data is given to it. The client will send the input data wrapped in an HTTP request object for which the prediction has to be made, and the API hosted on the server will extract the input data from the Flask request object, and then it will process the data. Then the processed data will be passed to the machine learning model. A pickle persisted model here is loaded to make predictions. Once the predictions is returned from the model, it will be sent back to the client wrapped in an HTTP response object.

VI. Conclusion

Diagnosis of Heart Disease Project and the work analyzed UCI Heart Disease Data Set to predict heart disease diagnosis, by utilizing the data analysis model/ algorithms, confusion matrix and the comparisons between models, as well as the optimization methods, in order to pursue a high accuracy as expected.

The project solution is implemented by Jupyter Notebook IDE, the language is Python with the imported libraries, such as Pandas, NumPy, matplotlib, %matplotlib inline, and other Python visualization libraries.

In the project the following model/ algorithms are implemented, Random Forest Classifier, Support Vector Machine, Logistic Regression, KNN Classifier, Naïve Bayes, Decision Tree, Ensemble Learning, Bagging algorithm.

A Machine Learning API is developed in the project by using Flask Microservice and pickle persisted model.

Acknowledgment

The work is a solely contributed to the project of Data Analysis (CP610, A). All permissions are under consideration and reserved.

The project is using Jupyter notebook. The version of the notebook server is: 6.3.0. Current Kernel Information: Python 3.9.4 (tags/v3.9.4:1f2e308, Apr 6 2021, 13:40:21) [MSC v.1928 64 bit (AMD64)]. IPython 7.23.1 -- An enhanced Interactive Python.

References

[1] Chaurasia, V. and Pal, S. (2013) Data Mining Approach to Detect Heart Disease. International Journal of Advanced Computer Science and Information Technology (IJACSIT), 2, 56-66. [Citation Time(s):1]

[2] UCI Machine Learning Repository, Heart Disease Data Set, http://archive.ics.uci.edu/ml/datasets/Heart+Disease

[3] Jiashu Zhao, *CP610 Data Analysis- Classification Basic Concep*, Week6B, 2021

[4] Data Mining: Concepts and Techniques, 2nd ed., Jiawei Han and Micheline Kamber, Morgan Kaufmann, 2006

[5] David Lapp, *Heart Disease Dataset*, Public Health Dataset, Kaggle, https://www.kaggle.com/johnsmith88/heart-disease-dataset, July 18, 2021

[6] IEEE, *Manuscript Templates for Conference Proceedings*, https://www.ieee.org/conferences/publishing/templates.html, July 18

[7] Data Mining: Concepts and Techniques, 2nd ed., Jiawei Han and Micheline Kamber, Morgan Kaufmann, 2006.

[8] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman, Mining of Massive Datasets (2nd Edition), Cambridge University Press, 2014.

[9] Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar, Introduction to Data Mining (2nd Edition), Pearson, 2018.

[10] D. J. Hand, H. Mannila, and P. Smyth, Principles of Data Mining, MIT Press, 2001.

Author's Biography

Dee Wu received the B.Sc. (computer software) from Heilongjiang University, now studies MCS program in Wilfrid Laurier University, working fulltime in TD Bank. As a Senior IT Developer, he works in system design, software development, technical support, and in charge code promotion, hot-fix, and non-critical incidents. As a Subject Matter Expert, he is specialized in SDLC, design pattern, methodologies, data warehouse/ ETL, database development/ administration, data mining/ machine learning solutions.