Loughborough University

COP528 Applied Machine Learning (LABs)

LAB Day 09. Ensemble learning

Introduction

The aim of this session is to understand several typical algorithms in Ensemble learning. The experiment includes **3** tutorial examples and **2** tasks. Please follow the tutorial to learn the basic functions and then complete the tasks.

Tutorial 01. Random Forest

This tutorial uses the Credit card default dataset to demonstrate the process of creating and parameterising a random forest. The following is a description of the Credit card default dataset and the task.

Data Set Information:

Credit card default dataset, containing 30,000 pieces of data, each with 23 attribute dimensions and a dichotomous label column, i.e., defaulted customer data or normal customer data.

Attribute Information:

This research employed a binary variable, default payment (Yes = 1, No = 0), as the response variable. This study reviewed the literature and used the following 23 variables as explanatory variables:

X1: Amount of the given credit (NT dollar): it includes both the individual consumer credit and his/her family (supplementary) credit.

X2: Gender (1 = male; 2 = female).

X3: Education (1 = graduate school; 2 = university; 3 = high school; 4 = others).

X4: Marital status (1 = married; 2 = single; 3 = others).

X5: Age (year).

X6 - X11: History of past payment. We tracked the past monthly payment records (from April to September, 2005) as follows: X6 = the repayment status in September, 2005; X7 = the repayment status in August, 2005; . . .;X11 = the repayment status in April, 2005. The measurement scale for

the repayment status is: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months; 9 = payment delay for nine months and above. X12-X17: Amount of bill statement (NT dollar). X12 = amount of bill statement in September, 2005; X13 = amount of bill statement in August, 2005; . . .; X17 = amount of bill statement in April, 2005. X18-X23: Amount of previous payment (NT dollar). X18 = amount paid in September, 2005; X19 = amount paid in August, 2005; . . .;X23 = amount paid in April, 2005.

Purpose:

To construct a random forest model based on credit card default data for classification prediction of new arrivals (in the example, the dataset is divided into data_train dataset and data_test dataset to simulate the training data and the new arrivals).

(1) Getting the data ready

Fristly, the dataset is divided into data_train and data_test to simulate the training data and the new arrivals.

```
#Getting the data ready
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
data = pd.read_csv('default of credit card clients.csv')
# print(data.head(3))
df = data.iloc[1:,1:24]
label=data.iloc[1:,-1]
#Random Forest

#Fristly, the dataset is divided into data_train and data_testto simulate the training data and the new arrivals
X_train, X_test, Y_train, Y_test = train_test_split(df,label,test_size=0.4)
```

(2) Let's first look at the comparison between decision trees and random forests (both with default parameters)

For data_train, it can often be divided into a train set for training the model and a validation set for evaluating the model during the training process.

```
#Random Forest
#parameters
# n_estimators: the number of decision tree models in the random forest model
# max_depth: the maximum depth of the decision tree model
# max_features: the maximum number of features to be selected when building the decision tree
# min_samples_leaf: the minimum number of samples of leaf nodes
# min_samples_split: the minimum number of samples allowed to be split by the current node
# criterion: the basis for node splitting

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import StratifiedShuffleSplit
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
import matplotlib.pyplot as plt
import time

# Let's first look at the comparison between decision trees and random forests (both with default parameters)
# For train data, it can often be divided into a train_set for training the model and a validation_set for evaluating
the selection of the model during the training process.
X_trainset, X_valset, Y_trainset, Y_valset = train_test_split(X_train,Y_train,test_size=0.3)
clf = DecisionTreeClassifier(random_state=0)
rfc = RandomForestClassifier(random_state=0)
clf = clf.fit(X_trainset,Y_trainset)
rfc = rfc.fit(X_trainset,Y_trainset)
score_c = clf.score(X_valset,Y_valset)
score_r = rfc.score(X_valset,Y_valset)
print("Single Tree:{}".format(score_c),"Random Forest:{}".format(score_r))
```

(3) Let's try to compare decision trees and random forest models (with n_estimators=25) by tenfold cross-validation.

```
rfc_l = []
clf_l = []
for i in range(10):
  rfc = RandomForestClassifier(n_estimators=25)
  rfc_s = cross_val_score(rfc,X_train,Y_train,cv=10).mean()
  rfc_l.append(rfc_s)
  clf = DecisionTreeClassifier()
  clf_s = cross_val_score(clf,X_train,Y_train,cv=10).mean()
  clf_l.append(clf_s)
plt.plot(range(1,11),rfc_l,label = "Random Forest")
plt.plot(range(1,11),clf_l,label = "Decision Tree")
plt.legend()
plt.show()
```

(4) Recalling the 6 parameters of the random forest we gave at the beginning, we give an example of parameter tuning.

```
#n_estimators
start=time.time()
scorel = []
for i in range(0,300,10): # Iteratively build RF models containing 0-300 decision trees for comparison

  rfc = RandomForestClassifier(n_estimators=i+1,n_jobs=-1,random_state=90)
  score = cross_val_score(rfc,X_train,Y_train,cv=10).mean()
  scorel.append(score)
print(f'{max(scorel)},the best n_estimators is {(scorel.index(max(scorel))*10)+1}')
# end=time.time()
# print('Running time: %s Seconds'%(end-start))
plt.figure(figsize=[20,5])
plt.plot(range(1,301,10),scorel)
plt.show()
```

(5) The RF model contains decision trees with an optimum of 211 trees, which we narrowed down in order to determine again

```
scorel = []
for i in range(200,222):
  rfc = RandomForestClassifier(n_estimators=i+1,n_jobs=-1,random_state=90)
  score = cross_val_score(rfc,X_train,Y_train,cv=10).mean()
  scorel.append(score)
print(max(scorel),([*range(200,222)][scorel.index(max(scorel))]))
plt.figure(figsize=[20,5])
plt.plot(range(200,222),scorel)
plt.show()
```

(6) OK, we have used data_train (training set and val set) to select better parameters and now use it to make predictions on data_test.

```
rfc_best = RandomForestClassifier(n_estimators=211,n_jobs=-1,random_state=90)
rfc_best = rfc_best.fit(X_train,Y_train)
rfc_best.score(X_test,Y_test)
```

Tutorial 02. AdaBoost

Follow the dataset and tasks in tutorial01 to show the code of the AdaBoost algorithm.

(1) Getting the data ready

Fristly, the dataset is divided into data_train and data_test to simulate the training data and the new arrivals.

```
#Getting the data ready
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
data = pd.read_csv('default of credit card clients.csv')
# print(data.head(3))
df = data.iloc[1:,1:24]
label=data.iloc[1:,-1]
#Random Forest

#Fristly, the dataset is divided into data_train and data_testto simulate the training data and the new arrivals
X_train, X_test, Y_train, Y_test = train_test_split(df,label,test_size=0.4)
```

(2) Let's first look at the code of the AdaBoost classifier (with default parameters).

```
import numpy as np
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier

X_trainset, X_valset, Y_trainset, Y_valset = train_test_split(X_train,Y_train,test_size=0.3)
bdt = AdaBoostClassifier(n_estimators=200, random_state=0)

bdt.fit(X_trainset, Y_trainset)
print(bdt.score(X_valset,Y_valset))
```

(3) Classification fitting is done using Adaboost based on decision trees. Here we have chosen the SAMME algorithm with up to 200 weak classifiers and a step size of 0.8. In practice, you may need to select the best parameters by cross-validating the tuning parameters.

```
import numpy as np
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
X_trainset, X_valset, Y_trainset, Y_valset = train_test_split(X_train,Y_train,test_size=0.3)
bdt = AdaBoostClassifier(DecisionTreeClassifier(max_depth=2, min_samples_split=20, min_samples_leaf=5),
                         algorithm="SAMME",
                         n_estimators=200, learning_rate=0.8)

bdt.fit(X_trainset, Y_trainset)
print(bdt.score(X_valset,Y_valset))
```

Tutorial 03. Stacking

(1) Getting the data ready

Fristly, the dataset is divided into data_train and data_test to simulate the training data and the new arrivals.

```
#Getting the data ready
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
data = pd.read_csv('default of credit card clients.csv')
# print(data.head(3))
df = data.iloc[1:,1:24]
label=data.iloc[1:,-1]
#Random Forest

#Fristly, the dataset is divided into data_train and data_testto simulate the training data and the new arrivals
X_train, X_test, Y_train, Y_test = train_test_split(df,label,test_size=0.4)
```

(2) Classification using staking algorithms based on random forest and kNN.

```
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import StackingClassifier
import numpy as np
from sklearn import neighbors


estimators = [
    ('RF', RandomForestClassifier(n_estimators=10, random_state=42)),
    ('kNN',neighbors.KNeighborsClassifier(n_neighbors=2, weights='distance')),
]
Sclf = StackingClassifier(
    estimators=estimators, final_estimator=LogisticRegression()
)

from sklearn.model_selection import train_test_split
X_trainsewt, X_valset, y_trainset, y_valset = train_test_split(
     X_train,Y_train)
Sclf.fit(X_trainset, y_trainset).score(X_valset, y_valset)
```

Task 01. Tutorial Reproduction

Study the contents of tutorials 01, 02 and 03, complete the appropriate tuning parameters yourself and finally compare the results of Random Forest, AdaBoost and Stacking on the test dataset.

Task 02. Stacking

Based on what you have learned in the past few days, please select 3 classification algorithms, and use the Stacking algorithm to perform a classification task on the given data set and adjust the parameters appropriately to improve the classification results.

Data Set Information:

This has been collected using direct questionnaires from the patients of Sylhet Diabetes Hospital in Sylhet, Bangladesh and approved by a doctor.

Attribute Information:

Age 1.20-65

Sex 1. Male, 2.Female

Polyuria 1.Yes, 2.No.

Polydipsia 1.Yes, 2.No.

sudden weight loss 1.Yes, 2.No.

weakness 1.Yes, 2.No.

Polyphagia 1.Yes, 2.No.

Genital thrush 1.Yes, 2.No.

visual blurring 1.Yes, 2.No.

Itching 1.Yes, 2.No.

Irritability 1.Yes, 2.No.

delayed healing 1.Yes, 2.No.

partial paresis 1.Yes, 2.No.

muscle sti

ness 1.Yes, 2.No.

Alopecia 1.Yes, 2.No.

Obesity 1.Yes, 2.No.

Class 1.Positive, 2.Negative.

Note that this dataset requires a pre-processing phase.

Objective:

Using 17 Attributes per sample, to predict whether the sample is likely to have early diabetes (positive or negative)

```
#Getting the data ready
import pandas as pd
import numpy as np
from sklearn.ensemble import StackingClassifier
data = pd.read_csv('diabetes_data_upload.csv')
```