Artificial Intelligence of Things

Programming Practice 1

Learning objective

- Be familiar with ML and data processing libraries in python.
- Use two preselected *Classification* Machine Learning Methods for data prediction.
- Choose a machine learning method by yourself that is suitable for the given application scenario and analyze the prediction results.
  - Give reasons to adopt the specific machine learning model.
  - Compare the prediction result with the ones of the two preselected methods.

Based on the given dataset:

1. Check the correlation between columns with the probability of diabetes.
2. Build models using **Decision Tree**, **Random Forest,** and **your chosen methods** for diabetes prediction.
3. Take a screenshot of each model’s training result.

Requirement

1.Source code as .ipynb format (35%)

2. 4 screenshots (all-in-one PDF file)

a.Print out `<dataset>.info()` after removing low correlation columns.

- Please provide the reason(s) for the exclusion. (15%)

b.Print out `model.score()` of training data and test data using **Decision Tree** method (20%)

c.Print out `model.score()` of training data and test data using **Random Forest** method (20%)

d.Rationally select (or design) an ML model for the analysis of diabetes prediction. Discuss your reason(s).

- Print out each value of the accuracy of training data and test data using your selected method (10%)

Zip your source code and PDF file named studentID_hw1.zip, and upload it to Moodle before **23:59 on 11/1 (Tue)**.

Environment Setup

- Use **Jupyter Notebook**

jupyter

Notebook

↗ 6.4.8

- Required Library
  - Seaborn
  - imblean
  - sklearn
  - Note : You can use "pip install <package-name>" to install required packages.

- Required Dataset
  - DiabetesDataset.csv (from Kaggle)

Definition of dataset

Diabetes_binary : 0 = no diabetes, 1 = diabetes

HighBPsort : 0 = no high BP, 1 = high BP

HighChol: 0 = no high cholesterol, 1 = high cholesterol

CholCheck: 0 = no cholesterol check, 1 = yes cholesterol check in 5 years

BMI : Body Mass Index

Smoker : 0 = no, 1 = yes

Stroke : (Ever told) you had a stroke. 0 = no, 1 = yes

HeartDiseaseorAttack : coronary heart disease (CHD) or myocardial infarction (MI).0 = no, 1 = yes

PhysActivity : physical activity in past 30 days - not including job. 0 = no, 1 = yes

Fruit : Consume Fruit 1 or more times per day. 0 = no, 1 = yes

Veggies : Consume Vegetables 1 or more times per day 0 = no, 1 = yes

HvyAlcoholConsump : (adult men >=14 drinks per week and adult women>=7 drinks per week).0 = no, 1 = yes

AnyHealthcare : Have any kind of health care coverage, including health insurance, prepaid plans such as HMO, etc. 0 = no, 1 = yes

NoDocbcCost : Was there a time in the past 12 months when you needed to see a doctor but could not because of cost? 0 = no, 1 = yes

GenHlth : Would you say that in general your health is: scale 1-5   1 = excellent, 2 = very good, 3 = good, 4 = fair, 5 = poor

MentHlth : days of poor mental health scale 1-30 days

PhysHlth : physical illness or injury days in past 30 days scale 1-30

DiffWalk : Do you have serious difficulty walking or climbing stairs? 0 = no, 1 = yes

Sex : 0 = female, 1 = male

Age : 13-level age category (_AGEG5YR see codebook) 1 = 18-24, 9 = 60-64, 13 = 80 or older

Education : Education level (EDUCA see codebook) scale 1-6   1 = Never attended school or only kindergarten, 2 = elementary etc.

Income : Income scale (INCOME2 see codebook) scale 1-8 1 = less than $10,000, 5 = less than $35,000, 8 = $75,000 or more

Import

```
import numpy as np //liner algebra

import pandas as pd //data processing
```

Decision Tree

```
from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, mean_squared_error

from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import train_test_split
```

Random Forest

```
from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import train_test_split
```

Check Dataset Correlation

Read the file

```
df1=pd.read_csv('../DiabetesDataset.csv')
```

Check the correlation between columns with the probability of diabetes.

```
df1.drop('Diabetes_binary', axis=1).corrwith(df1.Diabetes_binary).plot(kind='bar', grid=True, figsize=(20,
8)

, title="Correlation with Diabetes_binary",color="#119ef5");
```

Remove low correlation columns to improve model training results

```
df1.drop([?,?,...] , inplace=True , axis=1)
```

Show the result after removing low correlation columns(*Screenshot 1*)

```
df1.info()
```

(?:complete this code by yourself)

Oversampling

Check label distribution

```
df1['Diabetes_binary'].value_counts().plot(kind = 'bar', title = 'Label Distribution')

plt.show()
```

If the dataset is imbalanced, you need to balance it in order to get a better model

```
class_0 = df1[df1['Diabetes_binary'] == 0]

class_1 = df1[df1['Diabetes_binary'] == 1]

class_1_over = class_1.sample(len(class_0), replace=True)

df1_new = pd.concat([class_1_over, class_0], axis=0)

df1_new['Diabetes_binary'].value_counts().plot(kind='bar', title='Label Distribution after Oversampling')


plt.show()
```

Model

DecisionTree:

```
x = df1_new.drop('Diabetes_binary', axis = 1) # features

y = df1_new[['Diabetes_binary']] # labels

x_train ,x_test ,y_train ,y_test =train_test_split( x,y,train_size =0.8) # 0.8 for training,0.2 for test

model=DecisionTreeClassifier( max_depth=25)

model.fit( ? ,? ) # Use features and labels for training to fit the model

train_acc=model.score( ? ,? )# for training performance evaluation

test_acc=model.score( ? ,? )# Use testdataset to evaluate the performance of model

print(train_acc)

print(test_acc)
```

(*Screenshot 2*)

Model

RandomForest:

```
x = df1_new.drop('Diabetes_binary', axis = 1)  # features

y = df1_new[['Diabetes_binary']]  # labels

x_train ,x_test ,y_train ,y_test =train_test_split( x,y,train_size =0.8)# 0.8 for training,0.2 for test
model_1 = RandomForestClassifier(n_estimators = 300, criterion = 'entropy',min_samples_split=10,
random_state=0)

model_1.fit(?,?)# fitting the model on the train data


train_acc1 = model_1.score(?,?)# for training performance evaluation
test_acc1 = model_1.score( ? ,? )# Use testdataset to evaluate the performance of model

print(train_acc1)

print(test_acc1)
```

(*Screenshot 3*)