精算與大數據專題 (期末練習)

**I. Without Data preprocessing**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Accuracy | Precision | Recall |
| CART | 0.7532 | 0.6429 | 0.7759 |
| XGBoost | 0.7857 | 0.7193 | 0.7069 |
| CatBoost | 0.7987 | 0.7455 | 0.7069 |
| Random Forest | 0.7922 | 0.7500 | 0.6724 |

**II. With Data preprocessing (Standardized + Remove highly correlated variables)**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Accuracy | Precision | Recall |
| CART | 0.7792 | 0.7000 | 0.7241 |
| XGBoost | 0.7597 | 0.6721 | 0.7069 |
| CatBoost | 0.7727 | 0.7018 | 0.6897 |
| Random Forest | 0.7792 | 0.7069 | 0.7069 |

Coding

--------- R語言---------

# 匯入套件

library(mlbench)

# 讀取資料

data("PimaIndiansDiabetes")

dim(PimaIndiansDiabetes)

levels(PimaIndiansDiabetes$diabetes)

head(PimaIndiansDiabetes)

# 匯出資料

write.csv(PimaIndiansDiabetes, "data.csv")

--------- python ---------

# 匯入套件

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score

from sklearn.preprocessing import StandardScaler

from xgboost import XGBClassifier

from catboost import CatBoostClassifier

warnings.filterwarnings("ignore")

# 讀取資料

df = pd.read\_csv("data.csv")

df = df.iloc[:, 1:]

print(f"Shape of df: {df.shape}")

print(df.head())

# LabelEncoding

df["diabetes"] = [0 if x == "neg" else 1 for x in df["diabetes"].values]

# 分割訓練集以及驗證集

X = df.drop("diabetes", axis=1)

y = df["diabetes"]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    X, y, train\_size=0.8, random\_state=123

    )

print(f"Shape of X\_train: {X\_train.shape}")

print(f"Shape of y\_train: {y\_train.shape}")

print(f"Shape of X\_test: {X\_test.shape}")

print(f"Shape of y\_test: {y\_test.shape}")

# predict without data preprocessing

def display\_result(y\_true, y\_pred):

    print(f"ACC(test): {accuracy\_score(y\_true, y\_pred): .4f}")

    print(f"Precision(test): {precision\_score(y\_true, y\_pred): .4f}")

    print(f"Recall(test): {recall\_score(y\_true, y\_pred): .4f}")

DT = DecisionTreeClassifier()

model\_dt = DT.fit(X\_train, y\_train)

y\_hat\_dt = model\_dt.predict(X\_test)

display\_result(y\_true=y\_test, y\_pred=y\_hat\_dt)

print("-" \* 20)

XGB = XGBClassifier()

model\_xgb = XGB.fit(X\_train, y\_train)

y\_hat\_xgb = model\_xgb.predict(X\_test)

display\_result(y\_true=y\_test, y\_pred=y\_hat\_xgb)

print("-" \* 20)

CAT = CatBoostClassifier(verbose=0)

model\_cat = CAT.fit(X\_train, y\_train)

y\_hat\_cat = model\_cat.predict(X\_test)

display\_result(y\_true=y\_test, y\_pred=y\_hat\_cat)

print("-" \* 20)

RF = RandomForestClassifier()

model\_rf = RF.fit(X\_train, y\_train)

y\_hat\_rf = model\_rf.predict(X\_test)

display\_result(y\_true=y\_test, y\_pred=y\_hat\_rf)

# standardization

scaler = StandardScaler()

X\_train.iloc[:, :] = scaler.fit\_transform(X\_train)

X\_test.iloc[:, :] = scaler.transform(X\_test)

# heatmap

c\_matrix = df.corr()

plt.figure(figsize=(10, 10))

sns.heatmap(c\_matrix, annot=True)

plt.xticks(rotation=45)

plt.title("HeatMap", size=16)

plt.show()

# drop pregnant

try:

    X\_train = X\_train.drop("pregnant", axis=1)

    X\_test = X\_test.drop("pregnant", axis=1)

except:

    print("Already drop.")

# predict with data preprocessing

print("-" \* 20)

DT = DecisionTreeClassifier()

model\_dt = DT.fit(X\_train, y\_train)

y\_hat\_dt = model\_dt.predict(X\_test)

display\_result(y\_true=y\_test, y\_pred=y\_hat\_dt)

print("-" \* 20)

XGB = XGBClassifier()

model\_xgb = XGB.fit(X\_train, y\_train)

y\_hat\_xgb = model\_xgb.predict(X\_test)

display\_result(y\_true=y\_test, y\_pred=y\_hat\_xgb)

print("-" \* 20)

CAT = CatBoostClassifier(verbose=0)

model\_cat = CAT.fit(X\_train, y\_train)

y\_hat\_cat = model\_cat.predict(X\_test)

display\_result(y\_true=y\_test, y\_pred=y\_hat\_cat)

print("-" \* 20)

RF = RandomForestClassifier()

model\_rf = RF.fit(X\_train, y\_train)

y\_hat\_rf = model\_rf.predict(X\_test)

display\_result(y\_true=y\_test, y\_pred=y\_hat\_rf)