МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

ЗВІТ

З ЛАБОРАТОРНОЇ РОБОТИ №4

З ДИСЦИПЛІНИ: « ПРОГРАМУВАННЯ ІНТЕЛЕКТУАЛЬНИХ ІНФОРМАЦІЙНИХ СИСТЕМ»

|  |  |
| --- | --- |
| Виконав:  ІП-15 Мєшков А.І. | Перевірив:  Курченко О.А. |

Київ 2023

ЗАВДАННЯ

Побудувати рендом форест звідси:

<https://www.kaggle.com/code/jhoward/how-random-forests-really-work/>

2.1. Натрейнити на датасеті звідси: '/kaggle/input/car-evaluation-data-set/car\_evaluation.csv'

encoder = ce.OrdinalEncoder(cols=['buying', 'maint', 'doors', 'persons', 'lug\_boot', 'safety'])

2.2 Вивести confusion matrix, auc, Classification report

3 Зробити буст попередньої моделі XGBoost. Порівняти результати <https://machinelearningmastery.com/random-forest-ensembles-with-xgboost/>

ХІД РОБОТИ

import numpy as np

import pandas as pd

import category\_encoders as ce

import seaborn as sns

from sklearn.ensemble import RandomForestClassifier

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

from sklearn.metrics import confusion\_matrix, roc\_auc\_score, classification\_report

import warnings

warnings.simplefilter("ignore")

data = pd.read\_csv('car\_evaluation.csv', names=['buying', 'maint', 'doors', 'persons', 'lug\_boot', 'safety', 'class'])

encoder = ce.OrdinalEncoder(cols=['buying', 'maint', 'doors', 'persons', 'lug\_boot', 'safety', 'class'])

data\_encoded = encoder.fit\_transform(data)

X = data\_encoded.drop('class', axis=1)

y = data\_encoded['class']

y = y-1

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

rf\_classifier = RandomForestClassifier()

rf\_classifier.fit(X\_train, y\_train)

y\_pred = rf\_classifier.predict(X\_test)

y\_prob = rf\_classifier.predict\_proba(X\_test)

auc = roc\_auc\_score(y\_test, y\_prob, multi\_class='ovr')

print("AUC:", auc)

class\_report = classification\_report(y\_test, y\_pred)

print("Classification Report:")

print(class\_report)

confusion\_mat = confusion\_matrix(y\_test, y\_pred)

print("Confusion Matrix:")

confusion = pd.DataFrame(data=confusion\_mat)

sns.heatmap(confusion, annot=True, fmt='d', cmap='YlGnBu')

**Output:**

**AUC: 0.9982936081597542**

**Classification Report:**

**precision recall f1-score support**

**0 0.99 1.00 1.00 235**

**1 0.99 0.92 0.95 83**

**2 0.88 0.82 0.85 17**

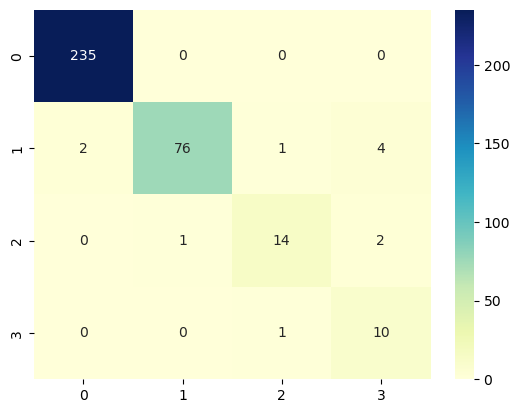
**3 0.62 0.91 0.74 11**

**accuracy 0.97 346**

**macro avg 0.87 0.91 0.88 346**

**weighted avg 0.97 0.97 0.97 346**

**Confusion Matrix:**



import os

os.environ['KMP\_DUPLICATE\_LIB\_OK']='True'

from xgboost import XGBClassifier

xgb\_classifier = XGBClassifier()

xgb\_classifier.fit(X\_train, y\_train)

y\_xgb\_pred = xgb\_classifier.predict(X\_test)

y\_xgb\_prob = xgb\_classifier.predict\_proba(X\_test)

auc\_xgb = roc\_auc\_score(y\_test, y\_xgb\_prob, multi\_class='ovr')

print("AUC for XGBoost:", auc\_xgb)

class\_report\_xgb = classification\_report(y\_test, y\_xgb\_pred)

print("Classification Report for XGBoost:")

print(class\_report\_xgb)

confusion\_mat = confusion\_matrix(y\_test, y\_xgb\_pred)

print("Confusion Matrix:")

confusion = pd.DataFrame(data=confusion\_mat)

sns.heatmap(confusion, annot=True, fmt='d', cmap='YlGnBu')

**Output:**

**AUC for XGBoost: 0.9993630179513603**

**Classification Report for XGBoost:**

**precision recall f1-score support**

**0 1.00 1.00 1.00 235**

**1 0.99 0.95 0.97 83**

**2 1.00 0.82 0.90 17**

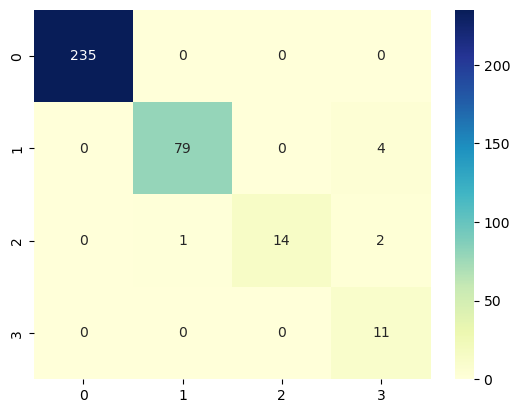
**3 0.65 1.00 0.79 11**

**accuracy 0.98 346**

**macro avg 0.91 0.94 0.91 346**

**weighted avg 0.99 0.98 0.98 346**

**Confusion Matrix:**



Висновок:

Обидві моделі виявилися дуже ефективними і точними, проте XGBoost має трохи більшу точність та кращі результати.