**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформатики та програмної інженерії**

**Звіт**

з лабораторної роботи №4 з дисципліни

«Програмування інтелектуальних інформаційних систем»

„Boosting”

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*ІП-11 Головня Олександр Ростиславович*

**Перевірив**

(прізвище, ім'я, по батькові)

*Баришич Л. М.*

Київ 2023

**Виконання:**

import pandas as pd

import seaborn as sns

!pip install category\_encoders

import matplotlib.pyplot as plt

from category\_encoders import OneHotEncoder

from sklearn.model\_selection import GridSearchCV

from sklearn.preprocessing import LabelEncoder

import warnings

warnings.filterwarnings('ignore')

from seaborn import heatmap

from sklearn.ensemble import RandomForestClassifier

import category\_encoders as ce

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

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

from xgboost import XGBRFClassifier

car\_df = pd.read\_csv("car\_evaluation.csv")

car\_df.columns = ['buying', 'meant', 'doors', 'persons', 'lug\_boot', 'safety', 'class']

category\_cols = ['buying', 'meant', 'doors', 'persons', 'lug\_boot', 'safety']

encoder = ce.OneHotEncoder(cols=category\_cols)

car\_df = encoder.fit\_transform(car\_df)

X = car\_df.drop(['class'], axis=1)

y = car\_df['class']

# Split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, train\_size=4/5, random\_state=42)

rf\_param\_grid = {

'n\_estimators': [i for i in range(100, 200, 50)],

'min\_samples\_leaf': [5, 10, 20],

'max\_features': ['log2', 'None', 'sqrt'],

'max\_depth': [4, 6, 10]

}

rf\_model = RandomForestClassifier(random\_state=42)

grid\_search = GridSearchCV(estimator=rf\_model, param\_grid=rf\_param\_grid,

cv=5, n\_jobs=-1, verbose=0, scoring='accuracy')

grid\_search.fit(X\_train, y\_train)

best\_parameters = grid\_search.best\_params\_

print(f"Best Parameters꞉ {best\_parameters}")

tuned\_rf\_model = RandomForestClassifier(max\_features='log2',

min\_samples\_leaf=10,

n\_estimators=100,

max\_depth=10,

random\_state=42)

tuned\_rf\_model.fit(X\_train, y\_train)

y\_pred\_rf = tuned\_rf\_model.predict(X\_test)

print(classification\_report(y\_test, y\_pred\_rf))

#####1

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

sns.heatmap(confusion\_matrix(y\_test, y\_pred\_rf), annot=True, fmt='d', cmap='Blues', cbar=False)

print(roc\_auc\_score(y\_test, tuned\_rf\_model.predict\_proba(X\_test), multi\_class='ovo'))

xgbrf\_model = XGBRFClassifier(random\_state=42)

xgbrf\_param\_grid = {

'n\_estimators': [i for i in range(100, 1000, 40)],

'max\_depth': [4, 6, 8, 10],

'min\_child\_weight': [1, 2, 4],

'colsample\_bytree': [0.5],

'learning\_rate': [0.01, 0.05, 0.1]}

grid\_search = GridSearchCV(estimator=xgbrf\_model, param\_grid=xgbrf\_param\_grid,

cv=5, n\_jobs=-1, verbose=0, scoring='accuracy')

label\_encoder = LabelEncoder()

y\_train\_encoded = label\_encoder.fit\_transform(y\_train)

y\_test\_encoded = label\_encoder.transform(y\_test)

grid\_search.fit(X\_train, y\_train\_encoded)

best\_parameters = grid\_search.best\_params\_

print(f"Best Parameters:{best\_parameters}")

tuned\_xgbrf\_model = XGBRFClassifier(n\_estimators=380, learning\_rate=0.01,

max\_depth=10,cosample\_bytree=0.5 ,min\_child\_weight=1)

tuned\_xgbrf\_model.fit(X\_train, y\_train\_encoded)

tuned\_xgbrf\_model.fit(X\_train, y\_train\_encoded)

y\_pred\_xgb = tuned\_xgbrf\_model.predict(X\_test)

print(classification\_report(y\_test\_encoded,y\_pred\_xgb))

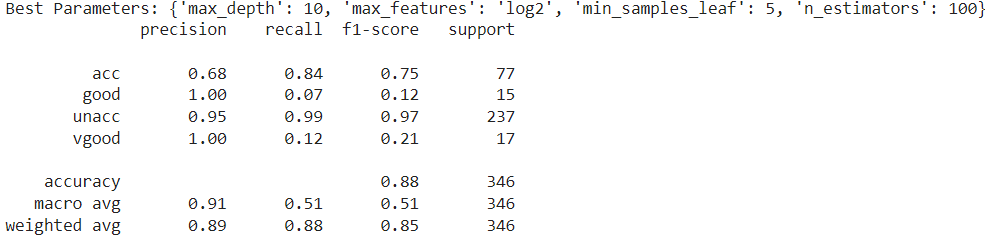
plt.subplot(1, 2, 2)

sns.heatmap(confusion\_matrix(y\_test\_encoded, y\_pred\_xgb), annot=True, fmt='d', cmap='Greens', cbar=False)

print(roc\_auc\_score(y\_test, tuned\_xgbrf\_model.predict\_proba(X\_test), multi\_class='ovo'))

1. **Тюнінг гіперпараметрів нашої модеᴫі**
2. **Виведення найкращих параметрів (**к-сть дерев, листів, предикторів,шлибина)

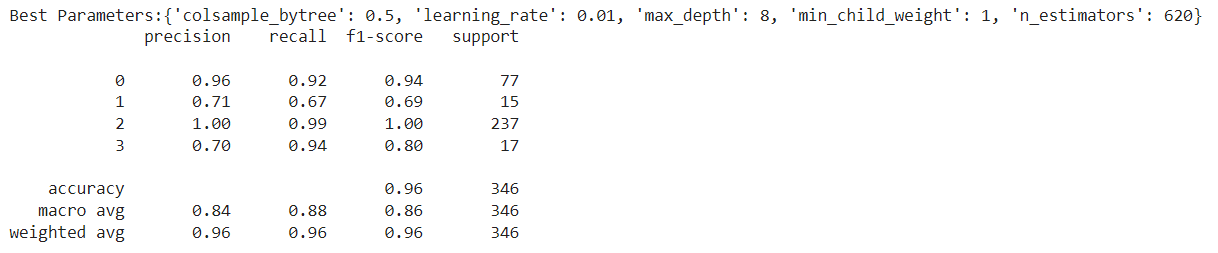
**Random Forest:**



**Roc\_auc\_score:**

****

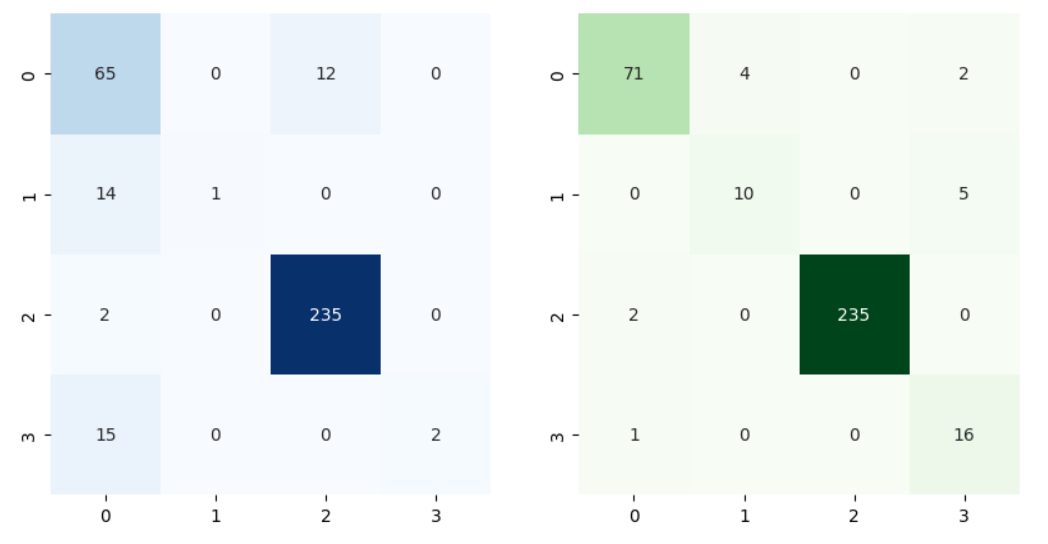
**XGboost:**

****

**Roc\_auc\_score:**

****

**Confusion matrixs:**

****