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')
```

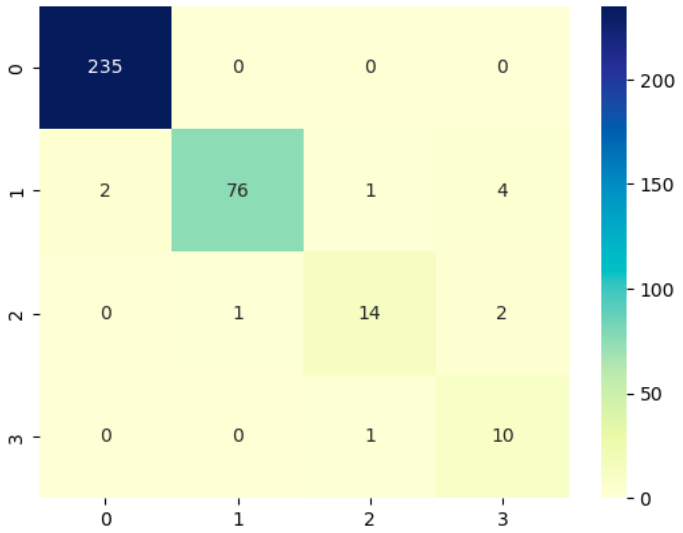
**Output:**

**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')
```

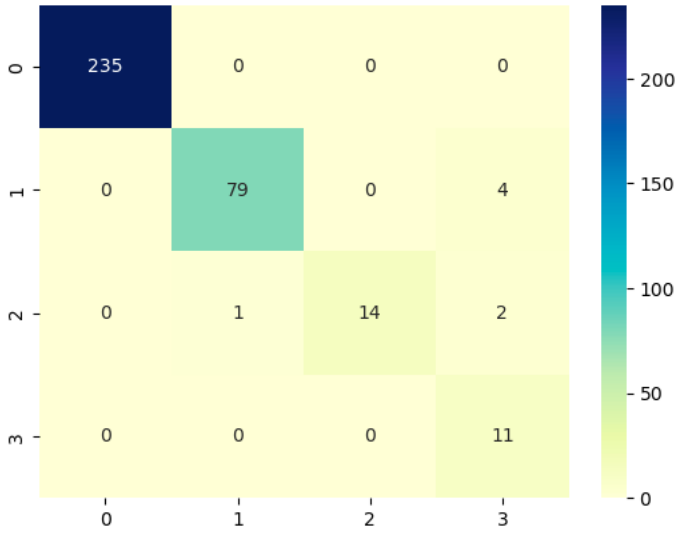
**Output:**

**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.