НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Лабораторна робота №3 з дисципліни

«Програмування інтелектуальних інформаційних систем»

Прийняв

аспірант кафедри ІПІ

Баришич Л.М.

Виконав

студент групи ІП-12

Казаков В.О.

“23” жовтня 2023 р.

Київ – 2023

Імпорт необхідних модулів:

from fastai.imports import \*

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.tree import plot\_tree

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

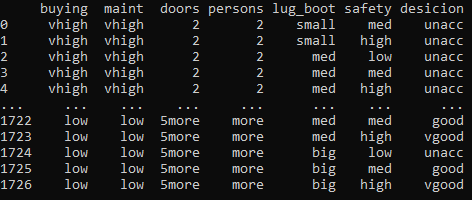
from xgboost import XGBRFClassifier

Імпорт датасету:

df = pd.read\_csv('car\_evaluation.csv')

df.columns = ['buying', 'maint', 'doors', 'persons', 'lug\_boot', 'safety', 'desicion']

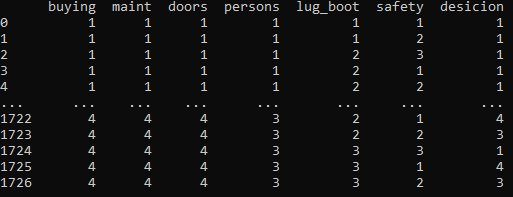
print(df)



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

df = encoder.fit\_transform(df)

print(df)



fig, axs = plt.subplots(2, 2, figsize=(11, 11))

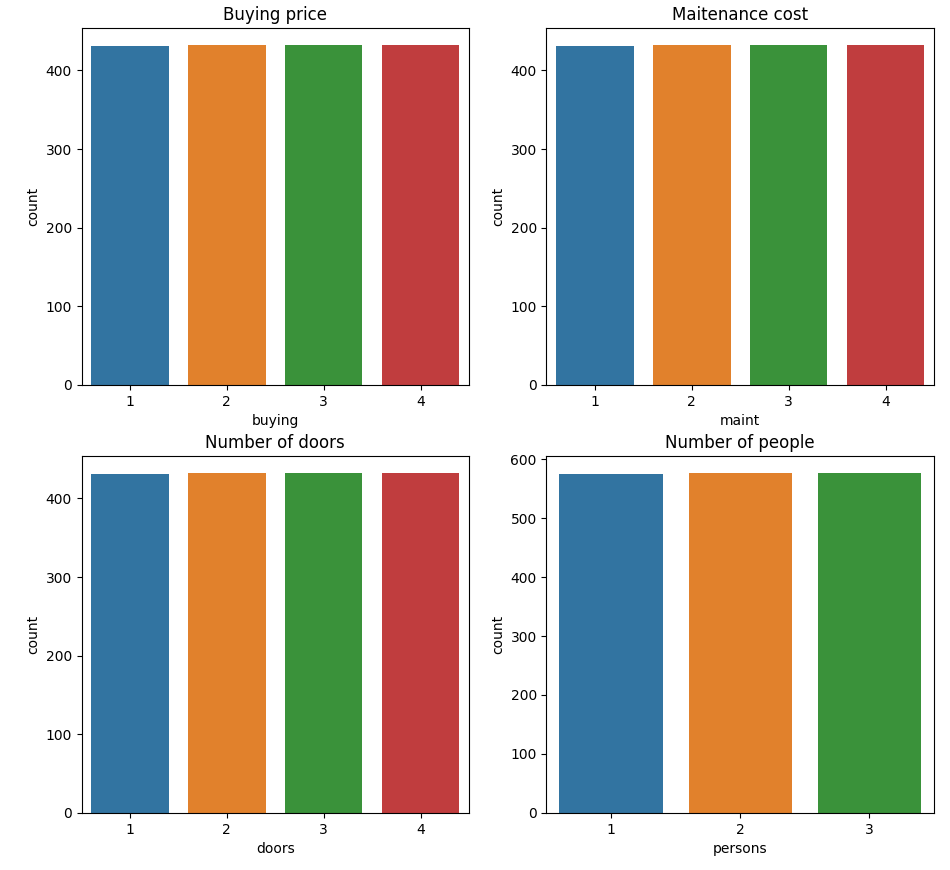
sns.countplot(data=df, x='buying', ax=axs[0][0]).set(title='Buying price')

sns.countplot(data=df, x='maint', ax=axs[0][1]).set(title='Maitenance cost')

sns.countplot(data=df, x='doors', ax=axs[1][0]).set(title='Number of doors')

sns.countplot(data=df, x='persons', ax=axs[1][1]).set(title='Number of people')

plt.show()



Random Forest:

x = df.drop('desicion', axis=1)

y = df['desicion'] - 1

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

rf = RandomForestClassifier(100, min\_samples\_leaf=5)

rf.fit(x\_train, y\_train)

print(f"Score: {rf.score(x\_test, y\_test)}")

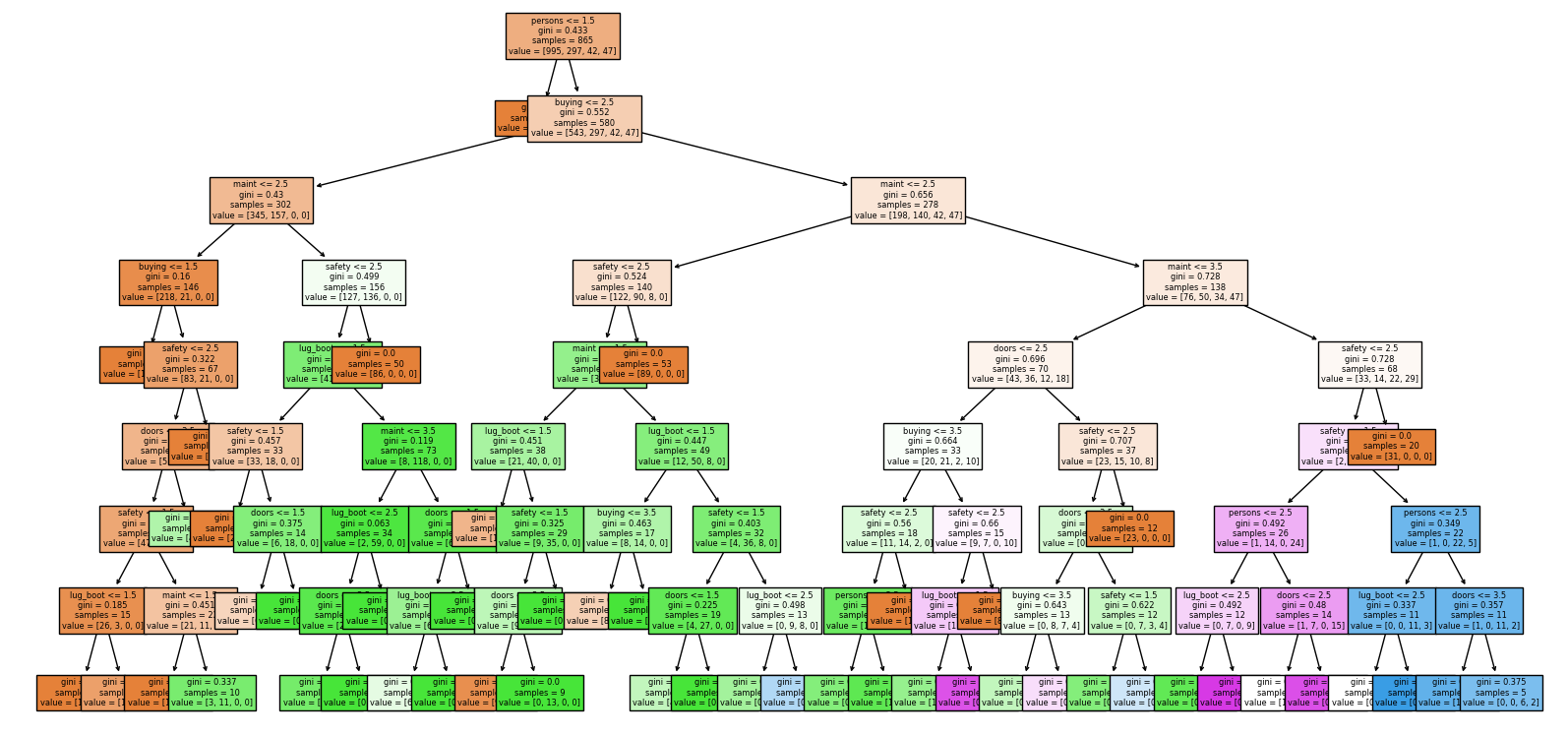
print(f"Mean absolute error: {mean\_absolute\_error(y\_test, rf.predict(x\_test))}")



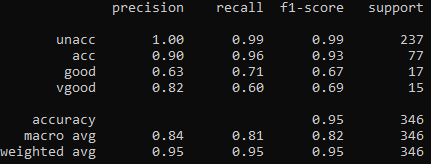
plt.figure(figsize=(50, 25))

plot\_tree(rf.estimators\_[0], filled=True, feature\_names=x\_train.columns)

plt.show()



print(classification\_report(y\_test, rf.predict(x\_test), target\_names=['unacc', 'acc', 'good', 'vgood']))



confusion = confusion\_matrix(y\_true=y\_test, y\_pred=rf.predict(x\_test))

fig, ax = plt.subplots()

im = ax.imshow(confusion)

ax.set\_xticks(np.arange(4), labels=['unacc', 'acc', 'good', 'vgood'])

ax.set\_yticks(np.arange(4), labels=['unacc', 'acc', 'good', 'vgood'])

plt.setp(ax.get\_xticklabels(), rotation=45, ha="right", rotation\_mode="anchor")

for i in range(0, 4):

for j in range(0, 4):

text=ax.text(j, i, confusion[i, j], ha="center", color="w")

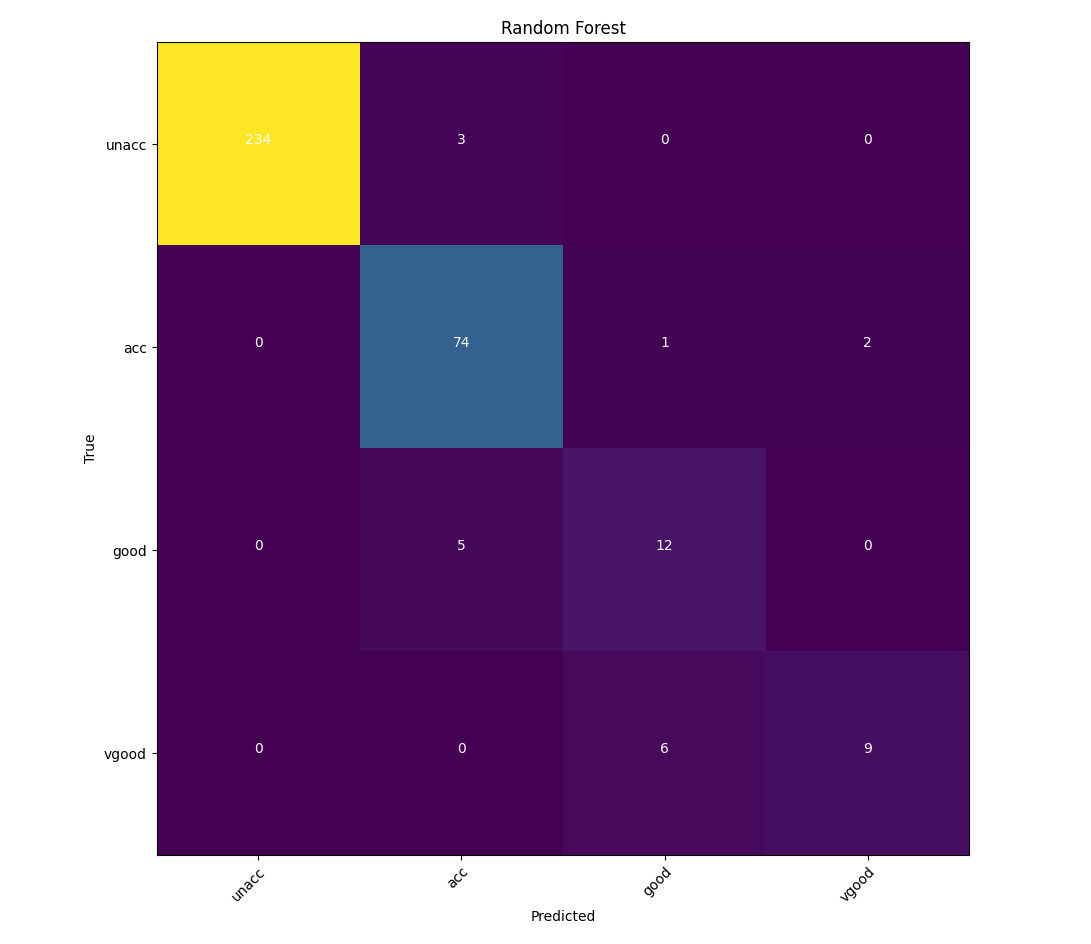
fig.tight\_layout()

plt.xlabel('Predicted')

plt.ylabel('True')

plt.title('Random Forest')

plt.show()



Balanced Random Forest:

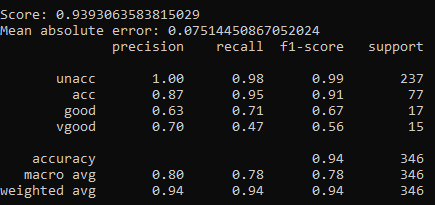
brf = XGBRFClassifier(n\_estimators=100, learning\_rate=0.5)

brf.fit(x\_train, y\_train)

print(f"Score: {brf.score(x\_test, y\_test)}")

print(f"Mean absolute error: {mean\_absolute\_error(y\_test, brf.predict(x\_test))}")

print(classification\_report(y\_test, brf.predict(x\_test), target\_names=['unacc', 'acc', 'good', 'vgood']))



confusion = confusion\_matrix(y\_true=y\_test, y\_pred=brf.predict(x\_test))

fig, ax = plt.subplots()

im = ax.imshow(confusion)

ax.set\_xticks(np.arange(4), labels=['unacc', 'acc', 'good', 'vgood'])

ax.set\_yticks(np.arange(4), labels=['unacc', 'acc', 'good', 'vgood'])

plt.setp(ax.get\_xticklabels(), rotation=45, ha="right", rotation\_mode="anchor")

for i in range(0, 4):

for j in range(0, 4):

text=ax.text(j, i, confusion[i, j], ha="center", color="w")

fig.tight\_layout()

plt.xlabel('Predicted')

plt.ylabel('True')

plt.title('Balanced Random Forest')

plt.show()

