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

Практикум №4

з курсу «Програмування інтелектуальних інформаційних систем»

на тему: «Аналітичні рішення за допомогою дерева рішень»

Викладач:
Курченко О.А.

Виконав:
Хільчук А.В.
студент 3 курсу
групи ІП-14 ФІОТ

Київ-2023

Практична робота №4

Тема: Аналітичні рішення за допомогою дерева рішень
Завдання:

1. Відповідно до заданого варіанту підготувати навчальну вибірку у вигляді таблиць MS Excel і зберегти їх як персональні файли. Для підготовки даних використовувати тематичні сайти Інтернет, результати проходження практик, довідники і каталоги.
2. Здійснити побудову дерева рішень і виконати його аналіз.
3. Отримати правила рішень і провести декілька експериментів.

Предметна бласть – передбачення захворювань печінки.

Виконання:

1. Відповідно до заданого варіанту підготувати необхідні дані у вигляді таблиць MS Excel і зберегти їх як персональні файли. Для підготовки даних використовувати тематичні сайти Інтернет, результати проходження практик, довідники і каталоги.

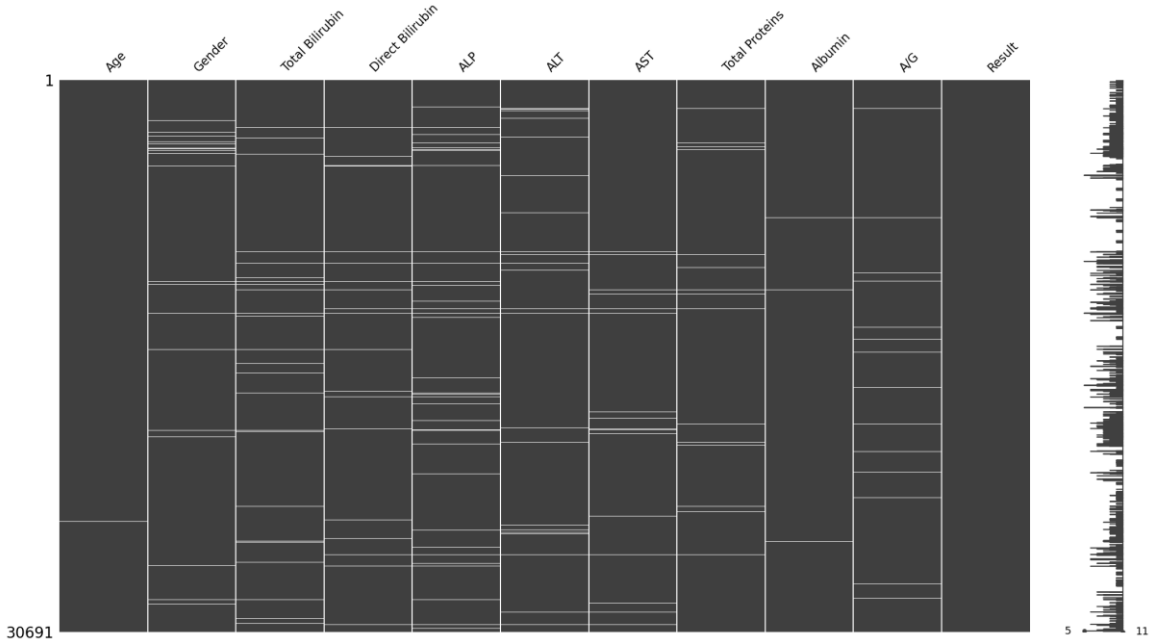
   Для виконання даної лабораторної роботи було взято даний набір даних:
   https://www.kaggle.com/datasets/abhi8923shriv/liver-disease-patient-dataset
   Даний датасет містить записи про відвувачів індійської лікарні, фіксуючі такі відомості: вік, стать пацієнта, його результати аналізів: загальний рівень білірубіну, прямий білірубін, рівні ензимів АЛТ, АСТ, АЛП, загальний білок крові, рівень альбуміну, відношення альбуміну до глобуліну та висновок, чи має в результаті пацієнт захворювання печінки, чи ні.

2. Здійснити побудову дерева рішень і виконати його аналіз.
   Код для даного розділу наведено в **додатку А.**
   Спочатку необхідно підготувати дані до аналізу.
   Перш за все, перйменуймо стовпці набору даних на більш лаконічні:

   - 'Age of the patient' - 'Age',
   - 'Gender of the patient' - 'Gender',
   - 'Alkphos Alkaline Phosphotase' - 'ALP',
   - 'Sgpt Alamine Aminotransferase' - 'ALT',

- 'Sgot Aspartate Aminotransferase' - 'AST',
- 'Total Protiens' - 'Total Proteins',
- 'ALB Albumin' - 'Albumin',
- 'A/G Ratio Albumin and Globulin Ratio' - 'A/G'

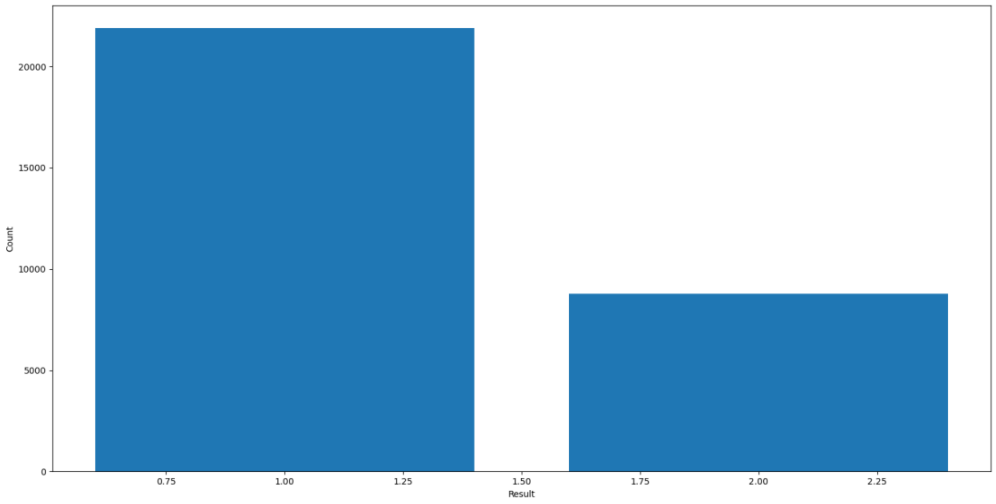
Проаналізуємо датафрейм на наявність відсутніх значень. Перш за все, побудуймо теплову карту відсутніх значень:



Як бачимо, відсутні значення наявні в 9 з 11 стовпців, однак наявні вони, зогляду, у невеликих кількостях і доволі часто відсутність полів рядка може збігатися для кількох стовпців. Проаналізуємо кількості появ відсутніх значень по стовпцях:

```
Total rows: 30691
Absent values per column:
    Age - 2
    Gender - 902
    Total Bilirubin - 648
    Direct Bilirubin - 561
    ALP - 796
    ALT - 538
    AST - 462
    Total Proteins - 463
    Albumin - 494
    A/G - 559
    Result - 0
```

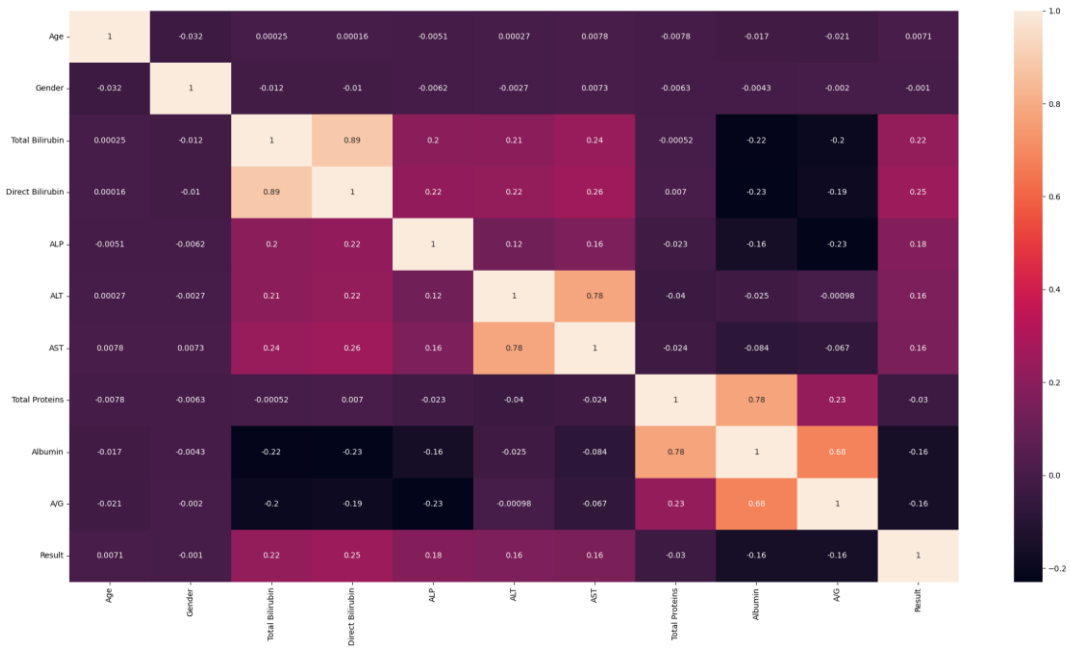
Усе таки, в Age також наявні два відсутні значення. У цілому, порівняно із загальним обсягом набору даних відсутніх значень незначний відсоток, отож було прийнято рішення рядки з відсутніми показниками просто видаляти. Після видалення, об’єм даних скоротився з 30691 запису до 27158 записів.
Візуалізуємо дані частоти значень стовпця Result:



Як бачимо, є значна незбалансованість класів. Двійка відображає те, що пацієнт виявився зі здоровою печінкою, а одиниця вказує на те, що захворювання є. Пояснюється така упередженість тим, що дані збиралися в лікарнях, а туди звертаються частіше пацієнти, що справді мають підстави підозрювати у себе захворювання печінки. Дану незбалансованість класів необхідно буде враховувати при подальших кроках виконання курсової роботи.

Також проведімо кодування деяких значень: у стовпці Gender перекодуємо Male та Female на 0 та 1 відповідно, а в Result замінимо 2 й й 1 на 0 та 1 відповідно, оскільки 1 вказує на те, що діагноз було поставлено, а 2 – що захворювання знайдено не було.

Тепер необхідно обрати предиктори. Для цього буде застосовано кореляційний аналіз:
Перш за все, будуємо матрицю кореляцій та відображаємо її у вигляді теплової карти:

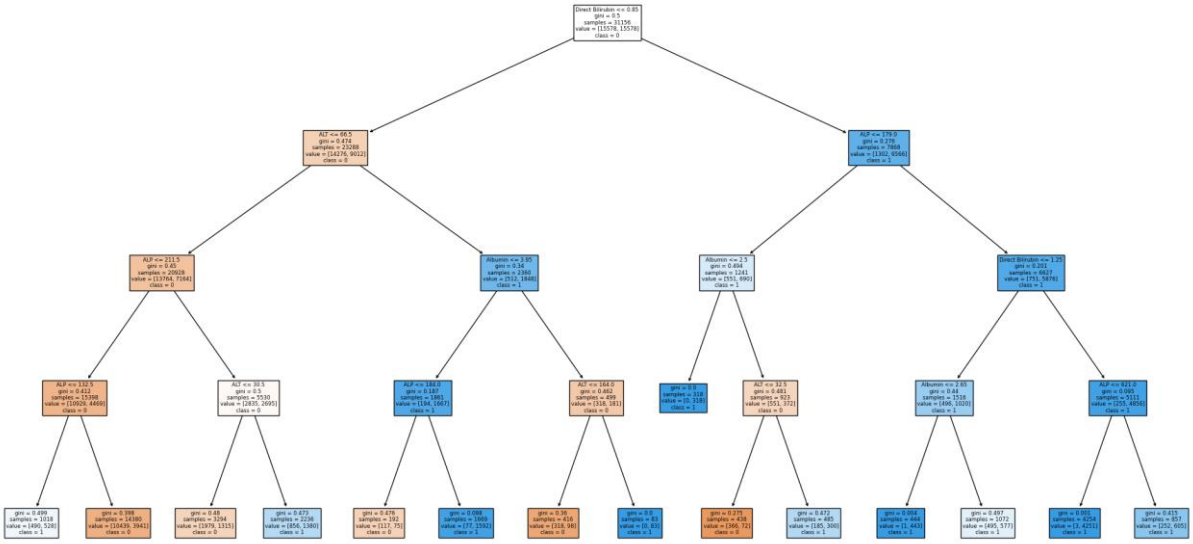


Тепер необхідно ліквідувати мультиколінеарність.
Загальний білірубін має сильну кореляцію з прямим білірубіному- видаляємо загальний білірубін. Аналогічно АЛТ сильно корелює з АСТ – видаляємо АСТ. Альбумін та співвідношення альбуміну до глобуліну теж мають неабияку кореляцію між собою. Очевидно, це через наявність альбуміну в даному відношенні. Спробуємо встановити рівень глобуліна звідси й встановити, чи впливатиме він на result.
Як бачимо, кореляція доволі слабка позитивна існує, однак вона все ж занадто далека від порогу 0.2, тому стовпець глобуліну, усе ж, буде відкинуто.
Отож, результуючий набір даних має наступний вигляд:

	Direct Bilirubin	ALP	ALT	Albumin	Result
0	0.1	187.0	16.0	3.3	1
1	5.5	699.0	64.0	3.2	1
2	4.1	490.0	60.0	3.3	1
3	0.4	182.0	14.0	3.4	1
4	2.0	195.0	27.0	2.4	1
...	...	...	...	...	...
30686	1.0	610.0	17.0	2.6	1
30687	1.3	482.0	22.0	2.4	1
30688	3.0	542.0	116.0	3.1	1
30689	1.0	231.0	16.0	1.6	1
30690	1.6	253.0	80.0	3.9	1

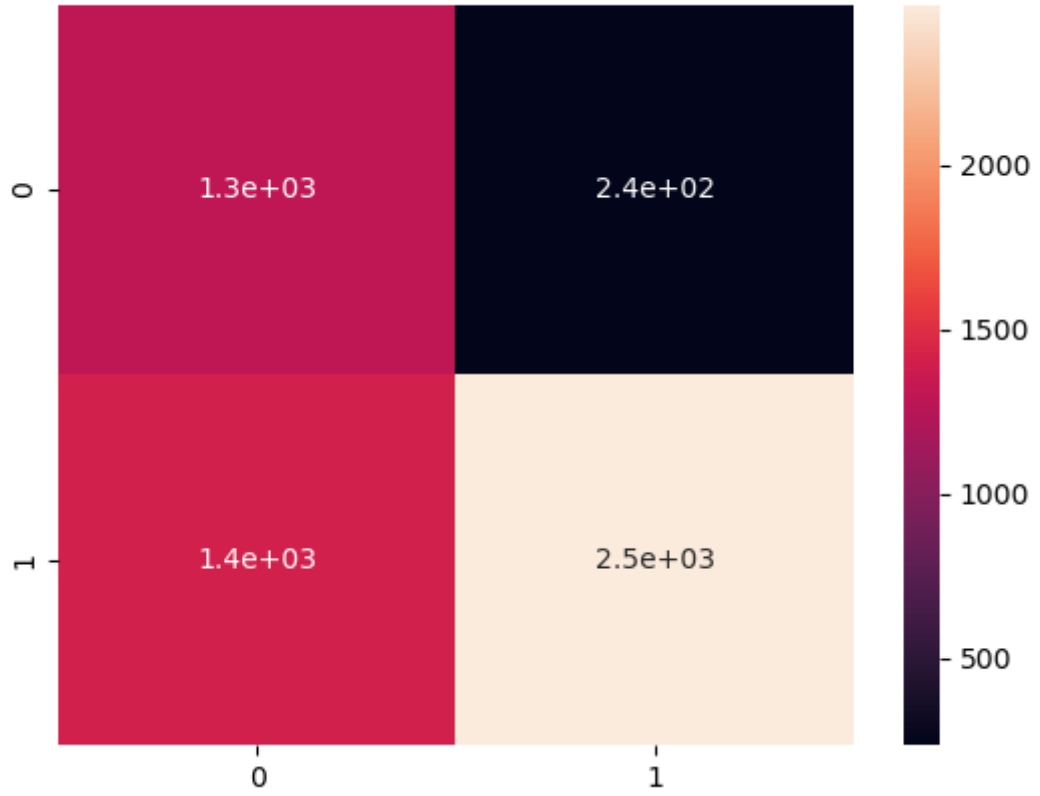
Отож, можна створювати дерево рішень.

Розділяємо вибірку на тренувальну та тестову, при цьому застосовубчи техніку оверсемплінгу, оскільки, як було підмічено вище, класи незбалансовані.
Тренуємо модель, обмеживши глибину в 4 вузли, і отримуємо дерево рішень наступного вигляду:



Для оцінки якості моделі, проаналізуємо такі метрики: точність, повнота й F-score, що знаходить компроміс між двома попередніми й обчислюється за формулою $\frac{2*\text{Accuracy}*\text{Recall}}{(\text{Accuracy}+\text{Recall})}$, а також звернемо увагу на матрицю невідповідностей, оскільки це важливо в контексті предметної області. Результати:

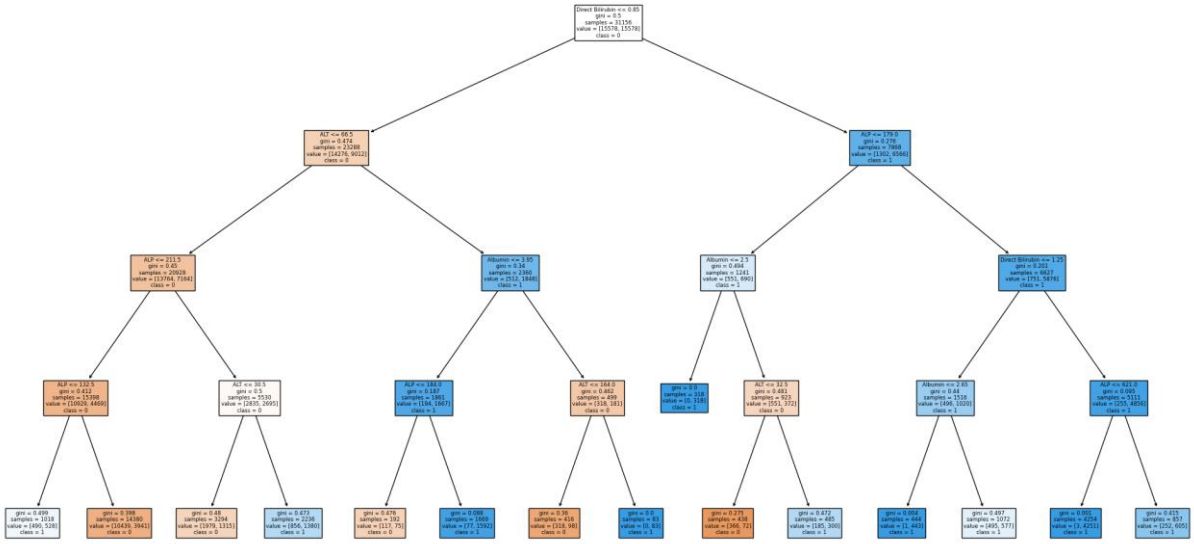
```
Accuracy: 0.696980854197349
Recall:0.6384615384615384
F-score:0.6664390330803913
```



Як бачимо, модель, попри доволі слабку кореляцію з параметрами, має точність і повноту більші статистичної вірогідності вгадування, однак у контексті предметної області вони замалі, щоб пропонувати повноцінне рішення.
Неприємлимим є те, що модель має надзвичайно велику кількість помилок 2 роду.

3. Отримати правила рішень і провести декілька експериментів.

Отож, правила рішення дерева є наступними:



Проводимо експерименти для 5 випадкових записів з тестового набору даних:

```
Experiment number1:
    Direct bilirubin: 0.3
    ALP: 498.0
    ALT: 28.0
    Albumin: 3.0
    Actual result: 1
    Predicted result: 0

Experiment number2:
    Direct bilirubin: 0.2
    ALP: 145.0
    ALT: 18.0
    Albumin: 3.9
    Actual result: 0
    Predicted result: 0

Experiment number3:
    Direct bilirubin: 0.3
    ALP: 190.0
    ALT: 20.0
    Albumin: 2.4
    Actual result: 1
    Predicted result: 0

Experiment number4:
    Direct bilirubin: 0.4
    ALP: 182.0
    ALT: 14.0
    Albumin: 3.4
    Actual result: 1
    Predicted result: 0

Experiment number5:
    Direct bilirubin: 0.1
    ALP: 265.0
    ALT: 97.0
    Albumin: 3.1
    Actual result: 1
    Predicted result: 1
```

Висновок

Отож, у ході виконання лабораторної роботи було взято набір даних про захворювання печінки пацієнтів індійської клініки, що фіксує вік, стать, а також результати аналізів, та підготовлено їх для подальшого аналізу. У рамках підготовки було перейменовано назви стовпців на більш лаконічні, видалено рядки з відсутніми значеннями та проведено кореляційний аналіз параметрів для відбору тих, що пов’язані з наявністю діагнозу. Опісля було побудовано дерево рішень і проаналізовано його точність, повноту, F-score, а також помилки другого та першого родів за допомогою матриці невідповідностей. Дані метрики виявилися недостатньо хорошими, що в контексті предметної області не дозволяє прогнозам моделі мати велику довіру. Урешті-решт було вибрано декілька записів з тестового набору даних та проведено наочний експеримент. Набуто практичних навичок створення дерев рішень мовою Python.

Додаток А

Текст скрипту для виконання лабораторної роботи

```
import pandas as pd
import missingno as msno
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, recall_score, confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from imblearn.over_sampling import SMOTE
from sklearn.tree import plot_tree
import random

if __name__=="__main__":
    pd.set_option('display.max_columns', None)

    data=pd.read_csv('data/Liver Patient Dataset
(LPD)_train.csv',encoding='cp1252')
    #Some of the column names have redundant spaces
    for col in data.columns:
        data.rename(columns={col: col.strip()}, inplace=True)

    #renaming columns into more terse alternatives
    data.rename(columns={'Age of the patient': 'Age',
                         'Gender of the patient':'Gender',
                         'Alkphos Alkaline Phosphotase':'ALP',
                         'Sgpt Alamine Aminotransferase':'ALT',
                         'Sgot Aspartate Aminotransferase':'AST',
                         'Total Protiens': 'Total Proteins',
                         'ALB Albumin':'Albumin',
                         'A/G Ratio Albumin and Globulin
Ratio':'A/G'},inplace=True)
    print(data)

    #removing NaN values
    msno.matrix(data)
    plt.show()

    print(f"\nTotal rows: {len(data)}\nAbsent values per column:")
    for col in data.columns:
        print(f"\t{col} - {data[col].isna().sum()}")


    print(f"\nProceeding to delete rows that contain NaN values\nRows before
deletion: {len(data)}")
    data.dropna(inplace=True)
    print(f"Rows after deletion:{len(data)}")

    counts = data['Result'].value_counts()
    labels = counts.index
    values = counts.values
    plt.bar(labels, values)
    plt.xlabel('Result')
    plt.ylabel('Count')
    plt.show()

    #some encoding
```

```
    #gender
    genders={'Male':0,'Female':1}
    data['Gender']=data['Gender'].map(genders).astype(int)

    #those with 2 tend to have normal enzymes level, so they are not likely
suffering from liver diseases
    #this mappping seems more logical
    results={2:0,1:1}
    data['Result']=data['Result'].map(results)

    #Determining predictors
    sns.heatmap(data.corr(),annot=True)
    plt.show()
    data.drop(columns=['Total Proteins','Age','Gender'],inplace=True)

    sns.heatmap(data.corr(),annot=True)
    plt.show()

    data['Globulin']=data['Albumin']/data['A/G']
    data.drop(columns=['Total Bilirubin','AST','A/G'],inplace=True)

    sns.heatmap(data.corr(),annot=True)
    plt.show()

    data.drop(columns=['Globulin'],inplace=True)
    print(data)


    #splitting data
    X_train, X_test, y_train, y_test = train_test_split(data.iloc[:, :-1],
data['Result'], test_size=0.2, random_state=42)



    # Create an instance of SMOTE
    oversampler = SMOTE(random_state=42)

    # Apply oversampling to the training data
    X_train_resampled, y_train_resampled = oversampler.fit_resample(X_train,
y_train)

    tree_clf = DecisionTreeClassifier(max_depth=4)
    tree_clf.fit(X_train_resampled, y_train_resampled)

    # Visualize the decision tree
    plt.figure(figsize=(20, 10))
    plot_tree(tree_clf, feature_names=X_train.columns,filled=True,
class_names=["0", "1"])
    plt.show()

    y_pred = tree_clf.predict(X_test)
    DTC_acc = accuracy_score(y_test, y_pred)
    DTC_recall = recall_score(y_test, y_pred)
    print(f'Metrics are:\n\tAccuracy: {DTC_acc}\n\tRecall:{DTC_recall}\n\tF-
score:{2 * DTC_acc * DTC_recall / (DTC_acc + DTC_recall)}')

    cm = confusion_matrix(y_test, y_pred)
    dtc_type2_err = cm[1, 0]
    dtc_type1_err = cm[0, 1]
```

```
    sns.heatmap(cm,annot=True)
    plt.grid(False)
    plt.show()



    #some experiments

    print("Performing some experiments:")
    for i in range(5):
        print(f"\nExperiment number{i+1}:")
        random_index = random.randint(0, len(X_test) - 1)

        sample_features = X_test.iloc[random_index]
        sample_label = y_test.iloc[random_index]

        print(f"\tDirect bilirubin: {sample_features['Direct Bilirubin']}")
        print(f"\tALP: {sample_features['ALP']}")
        print(f"\tALT: {sample_features['ALT']}")
        print(f"\tAlbumin: {sample_features['Albumin']}")
        print(f"\tActual result: {sample_label}")

        sample_prediction = tree_clf.predict([sample_features])[0]
        print(f"\tPredicted result: {sample_prediction}")
```