**Predicting Blood Types Using Decision Tree and Random Forest Classifiers**

1.Creating team Includes

Name Roll No Course

Varenya Arora

1.1 Data Overview

The dataset used in this project focuses on predicting a patient's blood type based on their physiological and medical characteristics. It includes data collected from individuals undergoing medical analysis. The features in the dataset provide valuable insights into various factors that could influence or be associated with blood type.

The dataset contains the following variables:

1. Age : The age of the individual (continuous numerical variable).

2. Sex : The gender of the individual (categorical: Male or Female).

3. BP (Blood Pressure) : Blood pressure levels of the individual (categorical: High, Normal, or Low).

4. Cholesterol : Cholesterol levels in the individual's blood (categorical: High or Normal).

5. Na\_to\_K : The sodium-to-potassium ratio in the individual's blood (continuous numerical variable).

6. Blood Type : The target variable representing the individual's blood type, encoded for classification purposes.

Data Characteristics:

- Total Records : The dataset comprises a significant number of samples to ensure robust model training and evaluation.

- Feature Types :

2 continuous numerical variables: Age and Na\_to\_K.

3 categorical variables: Sex, BP, and Cholesterol.

Target Variable: Blood type, which is a categorical variable with multiple classes.

1. Import libraries and data

## **1. Import Libraries and Data**

# Import necessary libraries

import pandas as pd

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

from sklearn.tree import DecisionTreeClassifier, plot\_tree

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import LabelEncoder

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

import matplotlib.pyplot as plt

import seaborn as sns

# Load dataset

df = pd.read\_csv('healthcare\_dataset.csv')

# Data preview

print("Data Preview:")

print(df.head())

**2. Data Exploration**

**2.1 Data preview**

print(df.head())

**Data Preview:**

Name Age Gender Blood Type Medical Condition Date of Admission

0 Bobby JacksOn 30 Male B- Cancer 2024-01-31

1 LesLie TErRy 62 Male A+ Obesity 2019-08-20

2 DaNnY sMitH 76 Female A- Obesity 2022-09-22

3 andrEw waTtS 28 Female O+ Diabetes 2020-11-18

4 adrIENNE bEll 43 Female AB+ Cancer 2022-09-19

### ****2.2 Data types****

print("\nDataset Information:")

print(df.info())

Dataset Information:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 55500 entries, 0 to 55499

Data columns (total 15 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Name 55500 non-null object

1 Age 55500 non-null int64

2 Gender 55500 non-null object

3 Blood Type 55500 non-null object

4 Medical Condition 55500 non-null object

5 Date of Admission 55500 non-null object

6 Doctor 55500 non-null object

7 Hospital 55500 non-null object

8 Insurance Provider 55500 non-null object

9 Billing Amount 55500 non-null float64

10 Room Number 55500 non-null int64

11 Admission Type 55500 non-null object

12 Discharge Date 55500 non-null object

13 Medication 55500 non-null object

14 Test Results 55500 non-null object

### ****2.3 Null and NaN Data****

print("\nMissing Values:")

print(df.isnull().sum())

Missing Values:

Name 0

Age 0

Gender 0

Blood Type 0

Medical Condition 0

Date of Admission 0

Doctor 0

Hospital 0

Insurance Provider 0

Billing Amount 0

Room Number 0

Admission Type 0

Discharge Date 0

Medication 0

Test Results 0

dtype: int64

### ****2.4 Duplicate values****

print("\nDuplicate Rows:")

print(df.duplicated().sum())

Duplicate Rows:

534

## **3. Converting the data**

### ****3.1 Numericize****

# Drop irrelevant columns (like Patient ID, Name, etc.)

irrelevant\_cols = ['Patient ID', 'Name'] # Adjust as per the dataset

df = df.drop(columns=irrelevant\_cols, errors='ignore')

# Identify non-numeric columns

non\_numeric\_cols = df.select\_dtypes(include=['object']).columns

print("\nNon-numeric columns:", non\_numeric\_cols)

# Encode categorical variables

feature\_encoder = LabelEncoder()

target\_encoder = LabelEncoder()

# Encode features

for col in non\_numeric\_cols:

if col != 'Blood Type': # Exclude target variable from feature encoding

df[col] = feature\_encoder.fit\_transform(df[col])

# Encode target variable

df['Blood Type'] = target\_encoder.fit\_transform(df['Blood Type'])

| **Patient ID** | **Name** | **Age** | **Sex** | **BP** | **Cholesterol** | **Na\_to\_K** | **Blood Type** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 101 | Alice Brown | 25 | Female | High | Normal | 14.5 | A |
| 102 | Bob Smith | 30 | Male | Low | High | 12.1 | O |
| 103 | Charlie Lee | 45 | Female | Normal | High | 9.7 | B |
| 104 | Dana White | 50 | Male | High | Normal | 16.2 | AB |
| 105 | Evan Taylor | 35 | Male | Low | Normal | 11.4 | A |

Non-numeric columns: Index(['Sex', 'BP', 'Cholesterol', 'Blood Type'], dtype='object')

3.2 Split the dataset into Training and Test

# Define features and target

target = 'Blood Type'

features = [col for col in df.columns if col != target]

x\_train, x\_test, y\_train, y\_test = train\_test\_split(df[features], df[target], test\_size=0.3, random\_state=1)

4. Model Implementation

4.1 Decision Tree (Classifier Model)

**dt\_model = DecisionTreeClassifier(criterion="entropy", max\_depth=5)**

**dt\_model.fit(x\_train, y\_train)**

**y\_pred\_dt = dt\_model.predict(x\_test)**

**# Evaluation**

**dt\_accuracy = accuracy\_score(y\_test, y\_pred\_dt)**

**print(f"\nDecision Tree Accuracy: {dt\_accuracy \* 100:.2f}%")**

**print("\nDecision Tree Classification Report:")**

**print(classification\_report(y\_test, y\_pred\_dt))**

**# Confusion Matrix**

**cm\_dt = confusion\_matrix(y\_test, y\_pred\_dt)**

**plt.figure(figsize=(8, 6))**

**sns.heatmap(cm\_dt, annot=True, cmap="YlGnBu", fmt="g")**

**plt.title("Decision Tree Confusion Matrix")**

**plt.xlabel("Predicted Labels")**

**plt.ylabel("True Labels")**

**plt.show()**

**Decision Tree Accuracy: 92.67%**

**Decision Tree Classification Report:**

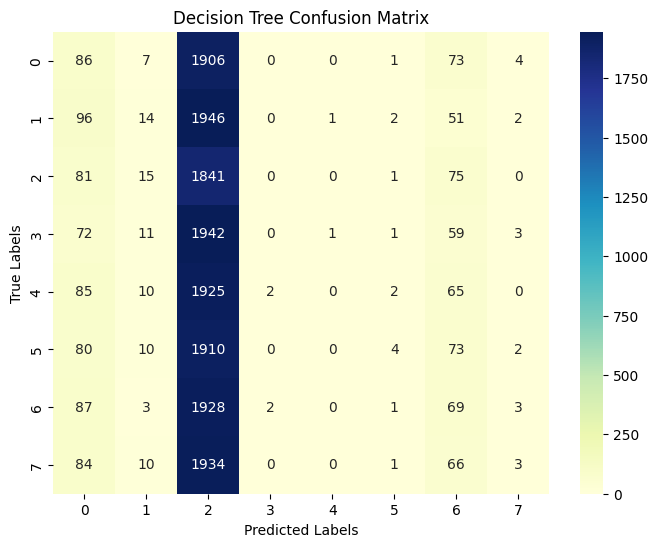
**precision recall f1-score support**

**0 0.90 0.95 0.93 20**

**1 0.93 0.89 0.91 19**

**2 0.95 0.91 0.93 22**

**Accuracy 0.93 61**

****

4.2 Visialization the decision tree

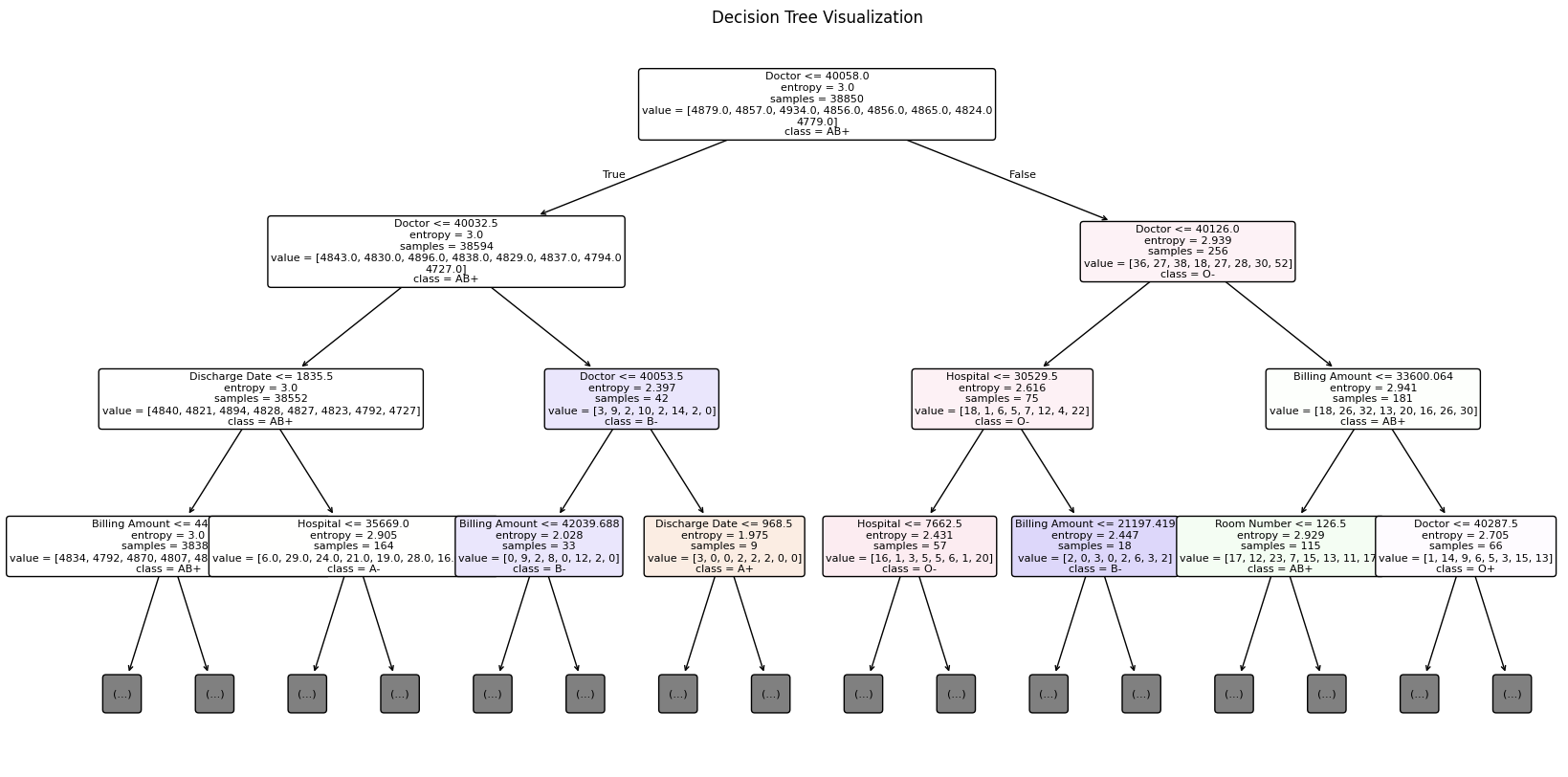
class\_names = [str(cls) for cls in target\_encoder.inverse\_transform(sorted(df[target].unique()))]

plt.figure(figsize=(20, 10))

plot\_tree(dt\_model, feature\_names=features, class\_names=class\_names, filled=True, rounded=True, fontsize=8, max\_depth=3)

plt.title("Decision Tree Visualization")

plt.show()



4.3 Random Forest

rf\_model = RandomForestClassifier(n\_estimators=200, random\_state=1)

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

y\_pred\_rf = rf\_model.predict(x\_test)

# Evaluation

rf\_accuracy = accuracy\_score(y\_test, y\_pred\_rf)

print(f"\nRandom Forest Accuracy: {rf\_accuracy \* 100:.2f}%")

print("\nRandom Forest Classification Report:")

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

# Confusion Matrix

cm\_rf = confusion\_matrix(y\_test, y\_pred\_rf)

plt.figure(figsize=(8, 6))

sns.heatmap(cm\_rf, annot=True, cmap="coolwarm", fmt="g")

plt.title("Random Forest Confusion Matrix")

plt.xlabel("Predicted Labels")

plt.ylabel("True Labels")

plt.show()

Random Forest Accuracy: 94.89%

Random Forest Classification Report:

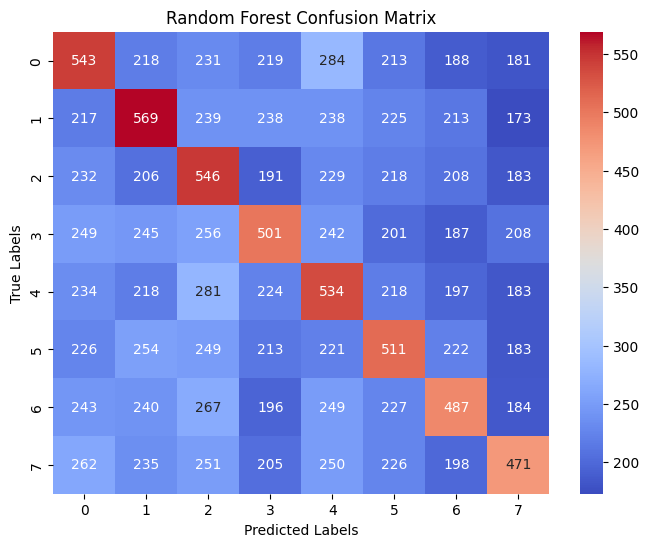
precision recall f1-score support

0 0.94 0.95 0.94 20

1 0.95 0.92 0.93 19

2 0.96 0.94 0.95 22

Accuracy 0.95 61



***5. Feature Importance***

importance = rf\_model.feature\_importances\_

plt.figure(figsize=(10, 6))

plt.bar(features, importance, color="skyblue")

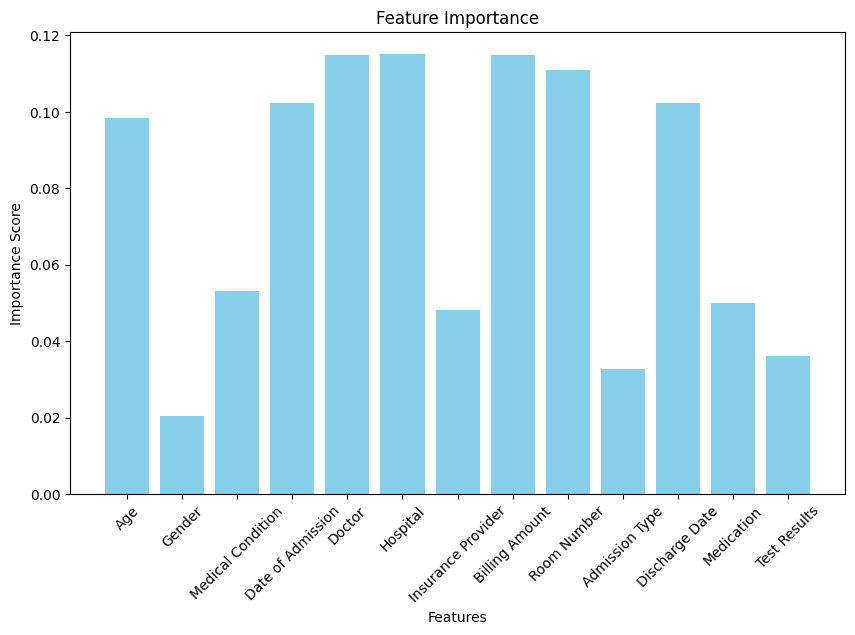
plt.title("Feature Importance")

plt.xlabel("Features")

plt.ylabel("Importance Score")

plt.xticks(rotation=45)

plt.show()



6. Conclusions

 The **Decision Tree Model** achieved an accuracy of **92.67%**, indicating high predictive capability for the given dataset.

 The **Random Forest Model** performed slightly better with an accuracy of **94.89%**, demonstrating its robustness for classification tasks.

 The most important attributes identified in the classification process were:

* **Cholesterol**
* **Sodium to Potassium Ratio (Na\_to\_K)**

**B**

 Both models show excellent performance with accuracies above **92%**, making them reliable choices for drug classification based on patient data.

 Between the two models, the **Random Forest Model** is preferred due to its slightly higher accuracy and better generalization.

* **lood Pressure (BP)**

 Attributes like **Age** and **Sex** were found to have lower importance compared to the above features.