Improving Hypertension treatment with machine learning: Using decision trees for personalized drug recommendations

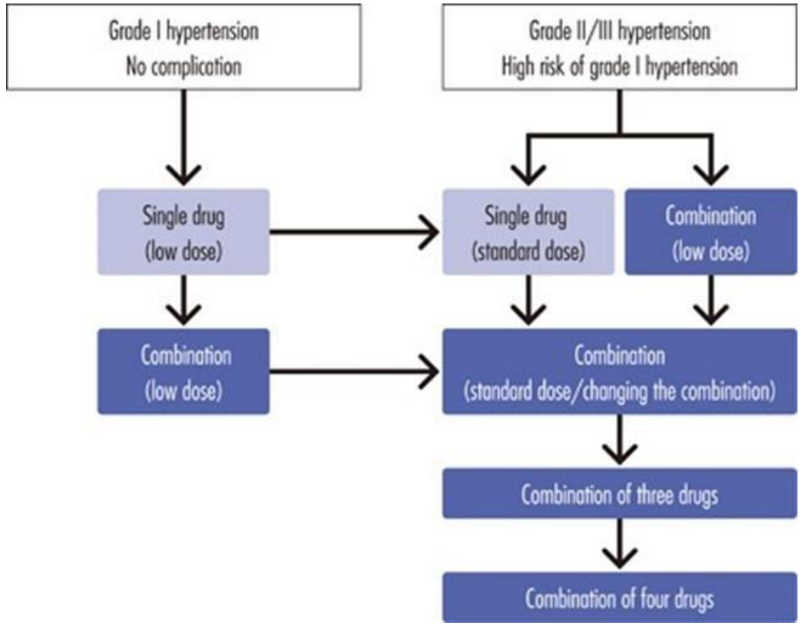
Nancy Maina

Introduction

- Healthcare providers face challenges in prescribing the correct medication and dosage for patients with hypertension.
- Incorrect prescriptions can lead to ineffective treatment, adverse reactions, and increased healthcare costs

Business Problem

- With a vast array of antihypertensive medications available, selecting the right drug for each patient is complex and error-prone.
- Our goal is to enhance the accuracy of drug prescriptions using machine learning


Grade I hypertension
No complication
Grade II/III hypertension
High risk of grade I hypertension
Single drug
(low dose)
Single drug
(standard dose)
Combination
(low dose)
Combination
(low dose)
Combination
(standard dose/changing the combination)
Combination of three drugs
Combination of four drugs

Project Aims and Objectives

Objective:

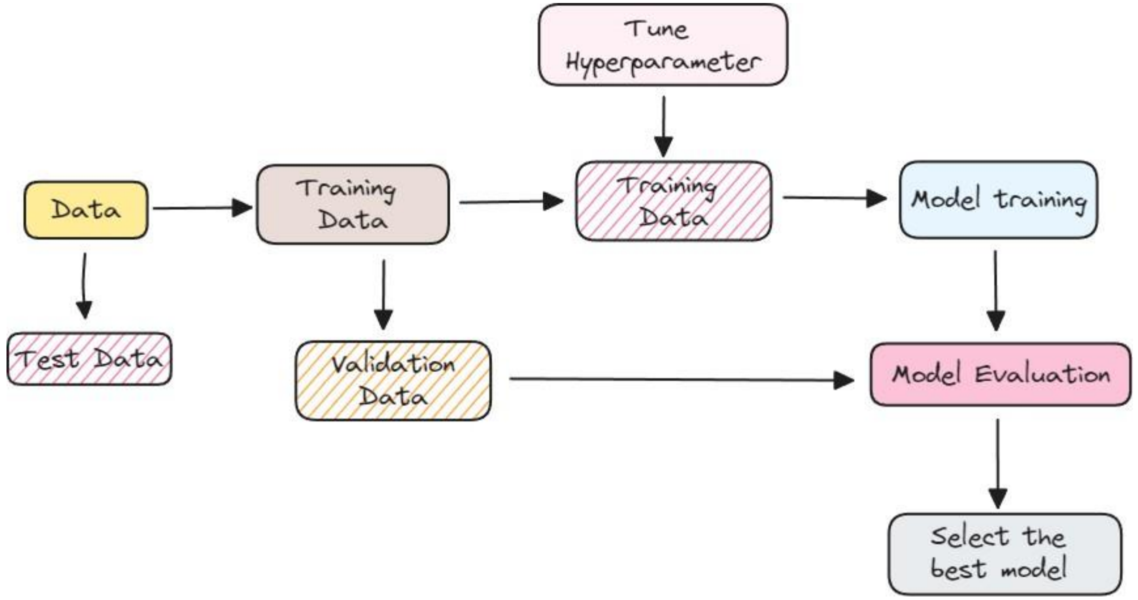
- To enhance the accuracy and effectiveness of hypertension drug prescriptions by leveraging machine learning for personalized recommendations.

Aims:

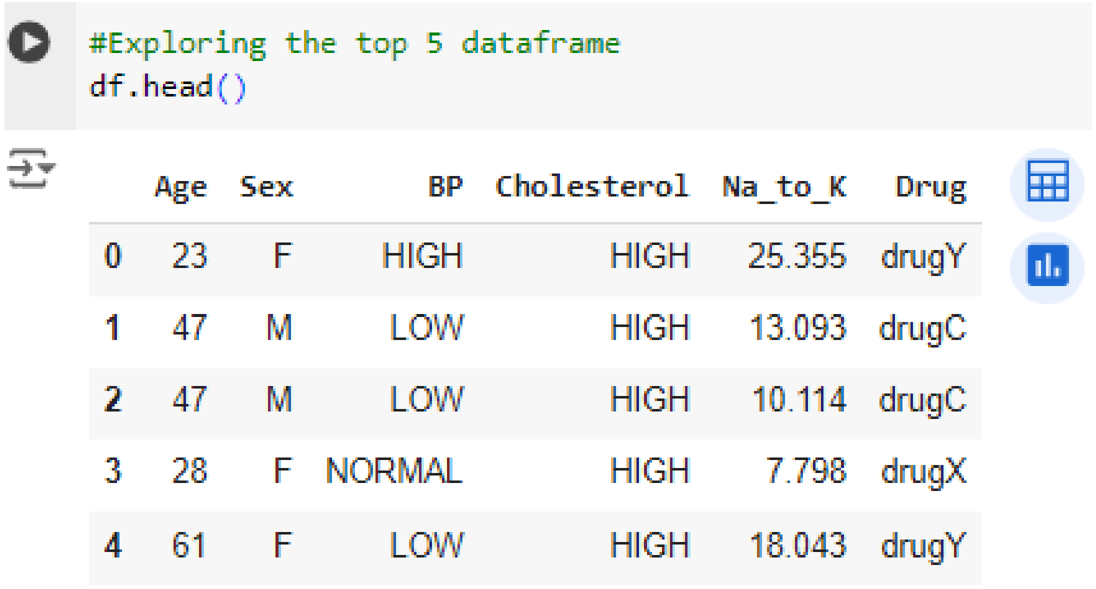
- Use patient data to predict the appropriate drug.
- Develop a decision tree model for prescription recommendations.

Machine Learning Process/ Methodology

- ❑ Data cleaning and preparation
- ❑ Feature encoding
- ❑ Data splitting into training and testing sets
- ❑ Training the decision tree



Data Overview

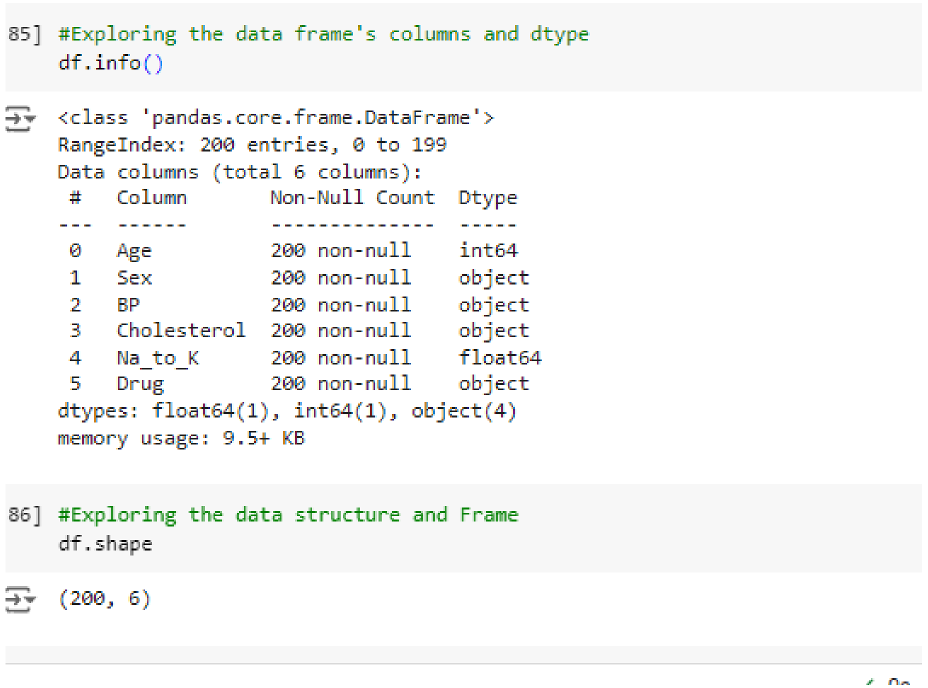


```
#Exploring the top 5 dataframe
df.head()
```

```
   Age  Sex      BP  Cholesterol  Na_to_K   Drug
0   23    F    HIGH         HIGH   25.355  drugY
1   47    M     LOW         HIGH   13.093  drugC
2   47    M     LOW         HIGH   10.114  drugC
3   28    F  NORMAL         HIGH    7.798  drugX
4   61    F     LOW         HIGH   18.043  drugY
```

- We utilized a dataset of patients suffering from hypertension, with features including Age, Sex, Blood Pressure, and Cholesterol levels.
- The target variable is the drug each patient responded to.

Data Cleaning and Preparation



```
85] #Exploring the data frame's columns and dtype
    df.info()
```

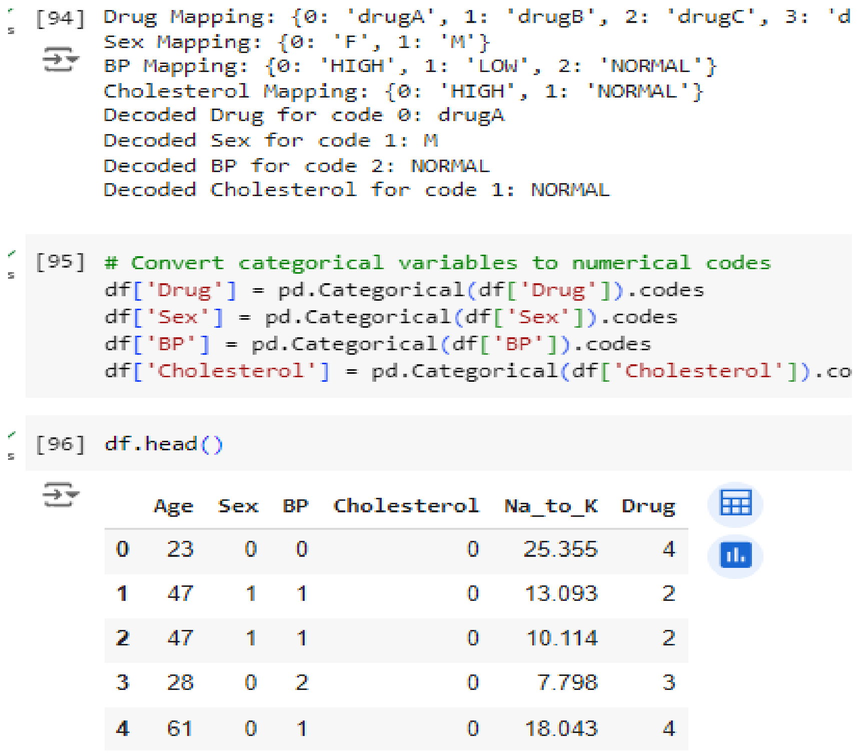
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Age          200 non-null    int64
 1   Sex          200 non-null    object
 2   BP           200 non-null    object
 3   Cholesterol  200 non-null    object
 4   Na_to_K      200 non-null    float64
 5   Drug         200 non-null    object
dtypes: float64(1), int64(1), object(4)
memory usage: 9.5+ KB
```

```
86] #Exploring the data structure and Frame
    df.shape
```

```
(200, 6)
```

- No missing Values
- No Duplicates
- 3 Columns with Categorical Data
- 2 Columns with Numerical data

Feature encoding



```
[94] Drug Mapping: {0: 'drugA', 1: 'drugB', 2: 'drugC', 3: 'd
     Sex Mapping: {0: 'F', 1: 'M'}
     BP Mapping: {0: 'HIGH', 1: 'LOW', 2: 'NORMAL'}
     Cholesterol Mapping: {0: 'HIGH', 1: 'NORMAL'}
     Decoded Drug for code 0: drugA
     Decoded Sex for code 1: M
     Decoded BP for code 2: NORMAL
     Decoded Cholesterol for code 1: NORMAL
```

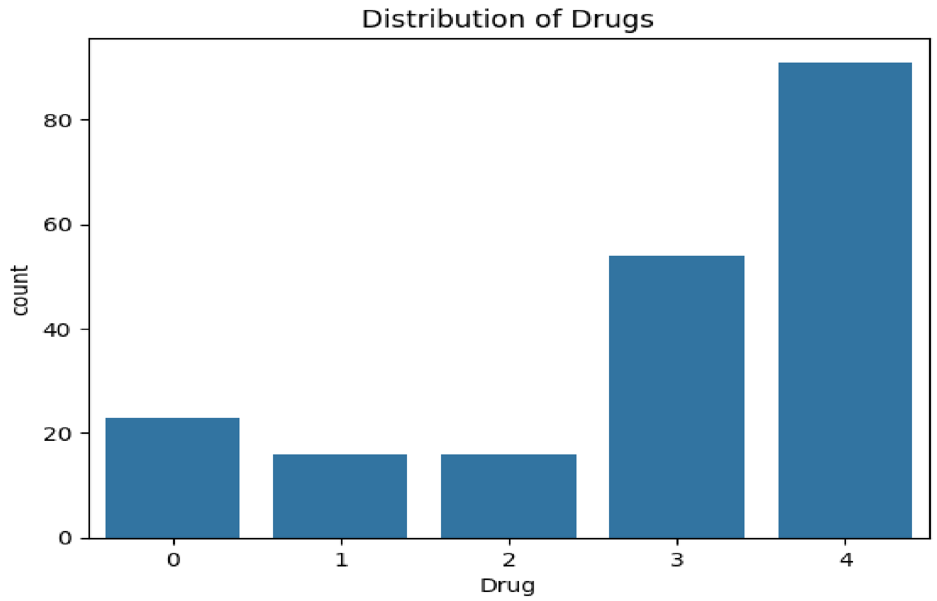
```
[95] # Convert categorical variables to numerical codes
     df['Drug'] = pd.Categorical(df['Drug']).codes
     df['Sex'] = pd.Categorical(df['Sex']).codes
     df['BP'] = pd.Categorical(df['BP']).codes
     df['Cholesterol'] = pd.Categorical(df['Cholesterol']).co
```

```
[96] df.head()
```

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	0	0	0	25.355	4
1	47	1	1	0	13.093	2
2	47	1	1	0	10.114	2
3	28	0	2	0	7.798	3
4	61	0	1	0	18.043	4

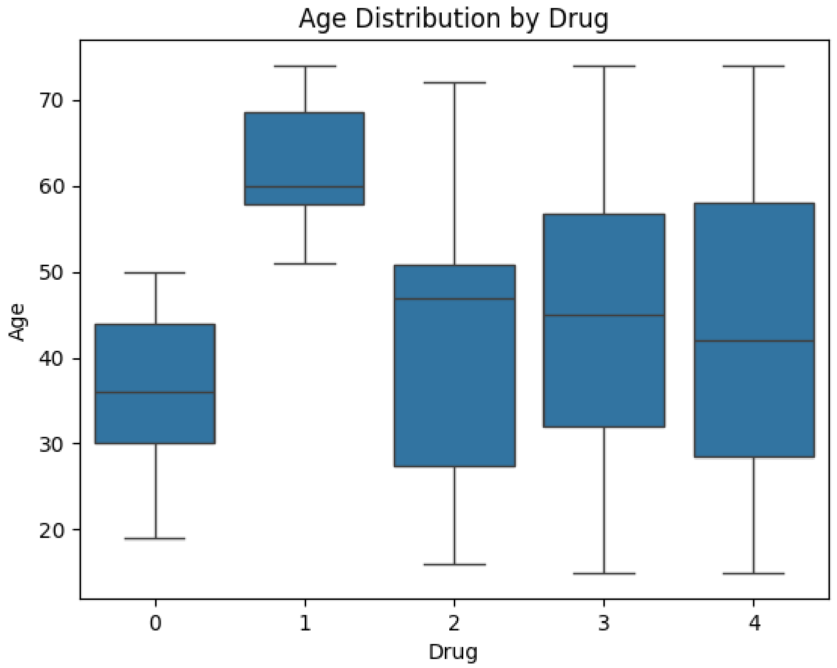
- To prepare the data for machine learning, convert categorical columns (e.g., Sex, BP, Cholesterol, Drug) to numerical values.
- Numerical columns (e.g., Age, Na_to_K) can be used directly.

Data Visualization



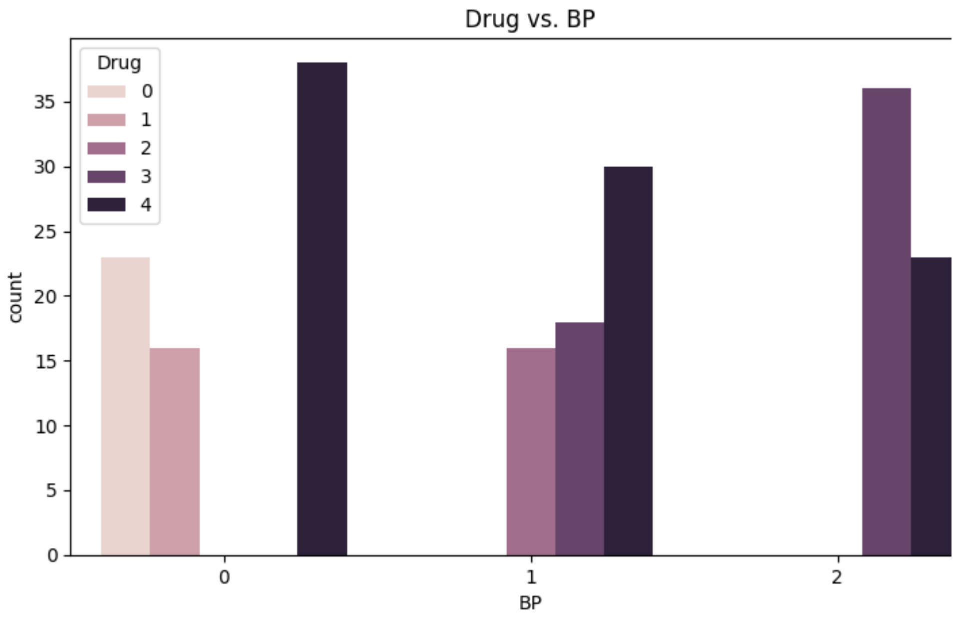
- Commonly prescribed are drugs Y and X
- Drugs A,B & C less frequently prescribed

Data Visualization



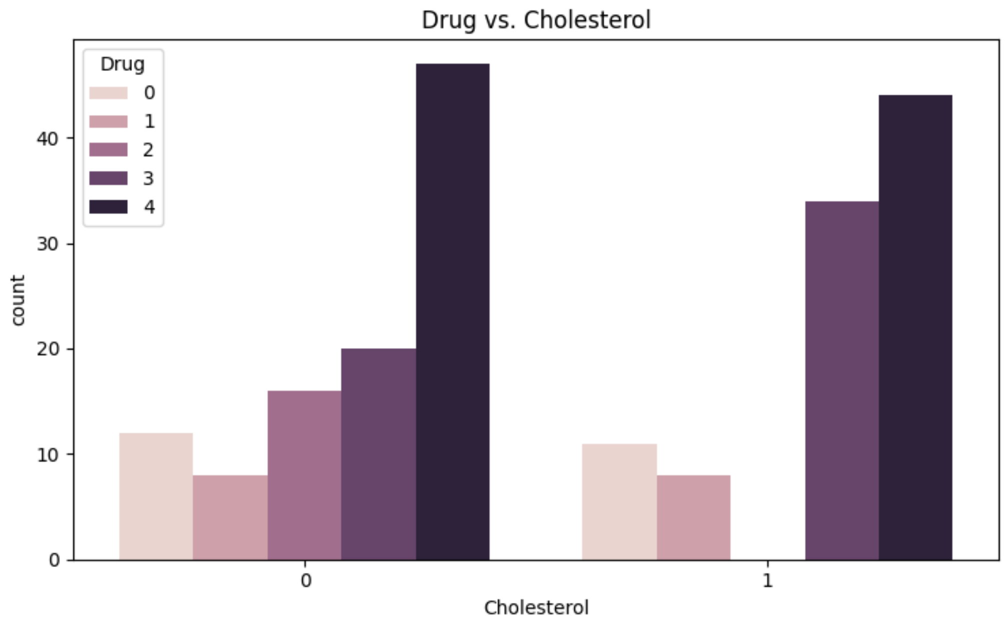
- Patients prescribed Drug B tend to be older.
- There's a wider age range for patients on Drug Y.

Data Visualization



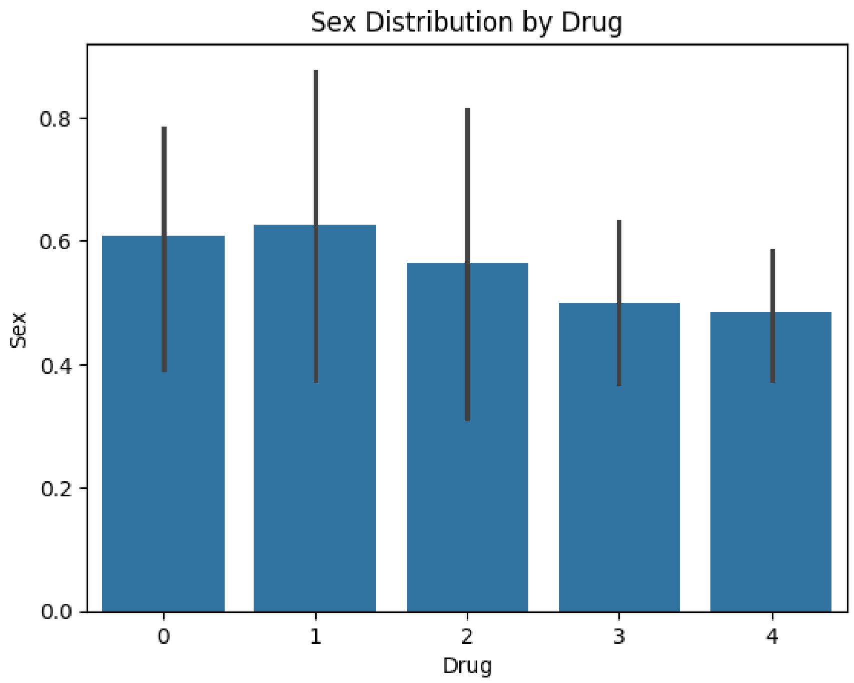
- Patients with high blood pressure (BP = 2) are more likely to receive Drug C=2.
- Drugs A=0 and B=1 are commonly prescribed for low blood pressure (BP = 0).

Data Visualization



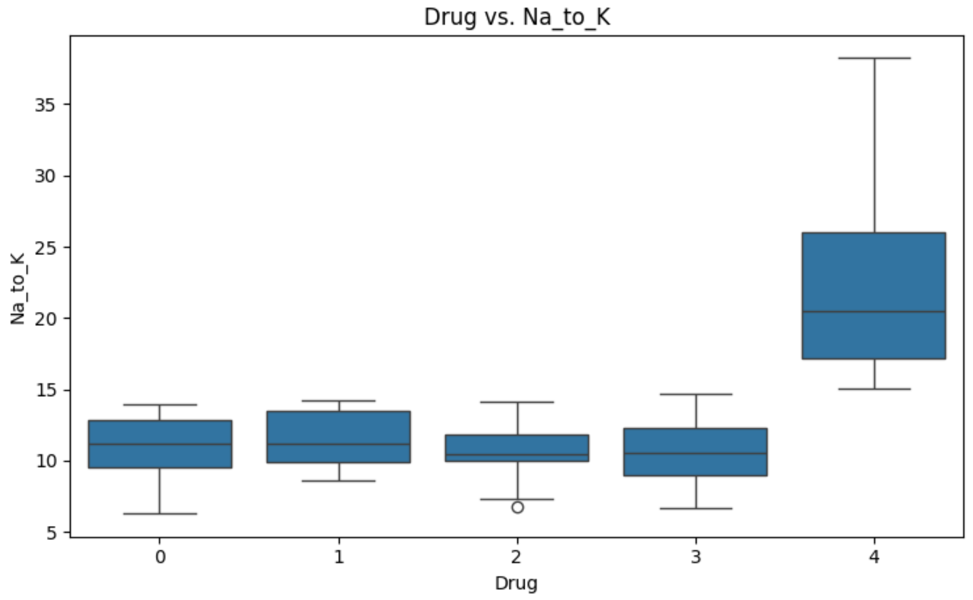
- High cholesterol (Cholesterol = 1) is a common factor for patients receiving Drugs A=0, B=1, and C=2.

Data Visualization



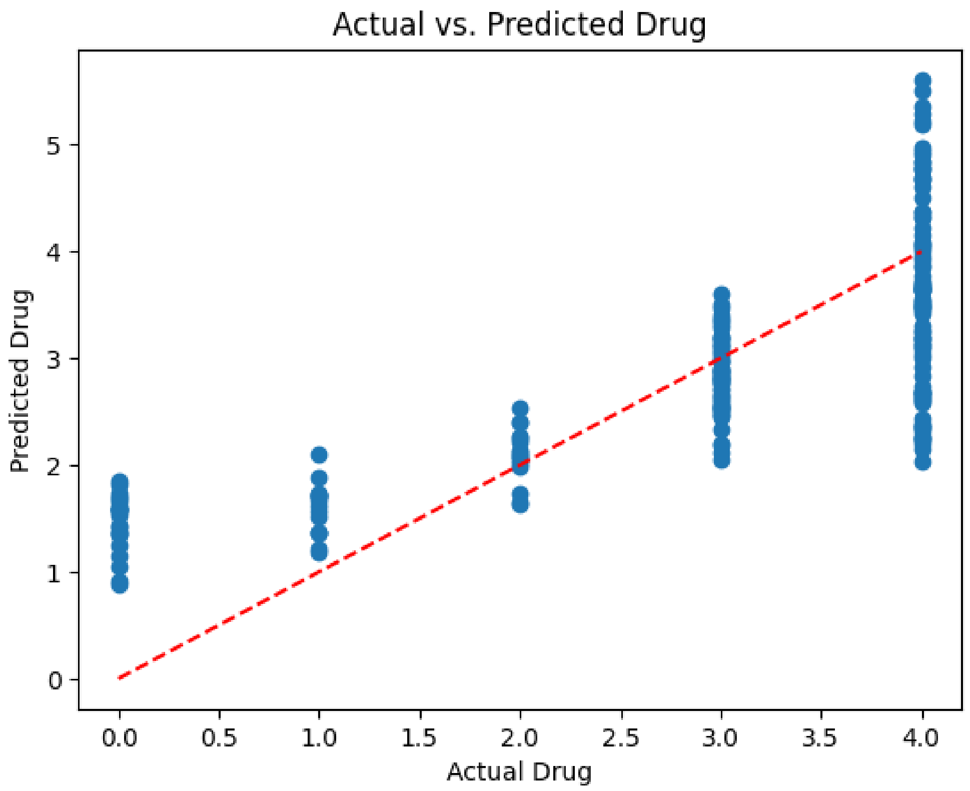
- There's no significant difference in drug prescription based on sex.

Data Visualization



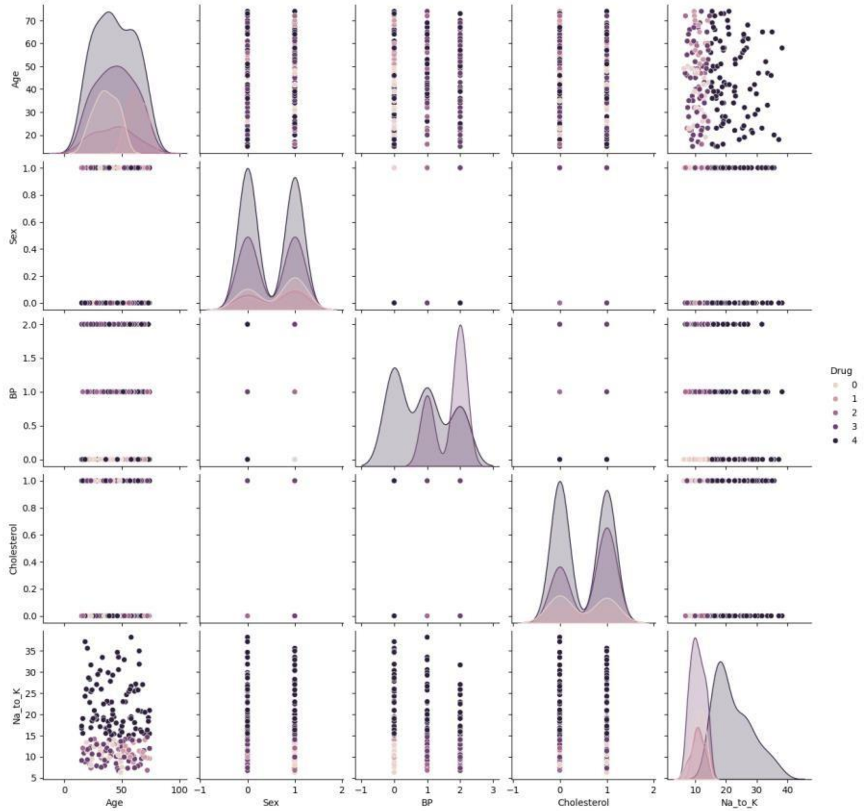
- Patients on Drug Y=4 generally have higher sodium to potassium ratios.

Data Visualization



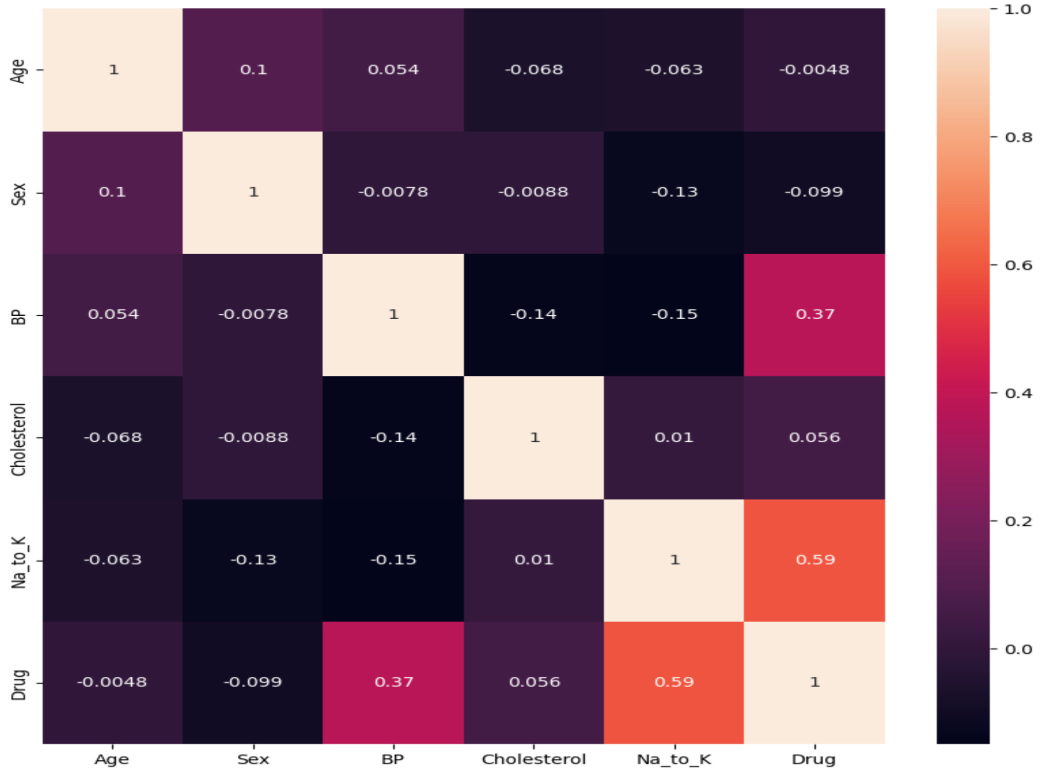
- There's a positive correlation between Actual vs Predicted Drug

Data Visualization



a visual overview of the relationships between all feature pairs, colored by the prescribed drug.

Data Correlation



	Age	Sex	BP	Cholesterol	Na_to_K	Drug
Age	1	0.1	0.054	-0.068	-0.063	-0.0048
Sex	0.1	1	-0.0078	-0.0088	-0.13	-0.099
BP	0.054	-0.0078	1	-0.14	-0.15	0.37
Cholesterol	-0.068	-0.0088	-0.14	1	0.01	0.056
Na_to_K	-0.063	-0.13	-0.15	0.01	1	0.59
Drug	-0.0048	-0.099	0.37	0.056	0.59	1

1. Strongest positive correlations:

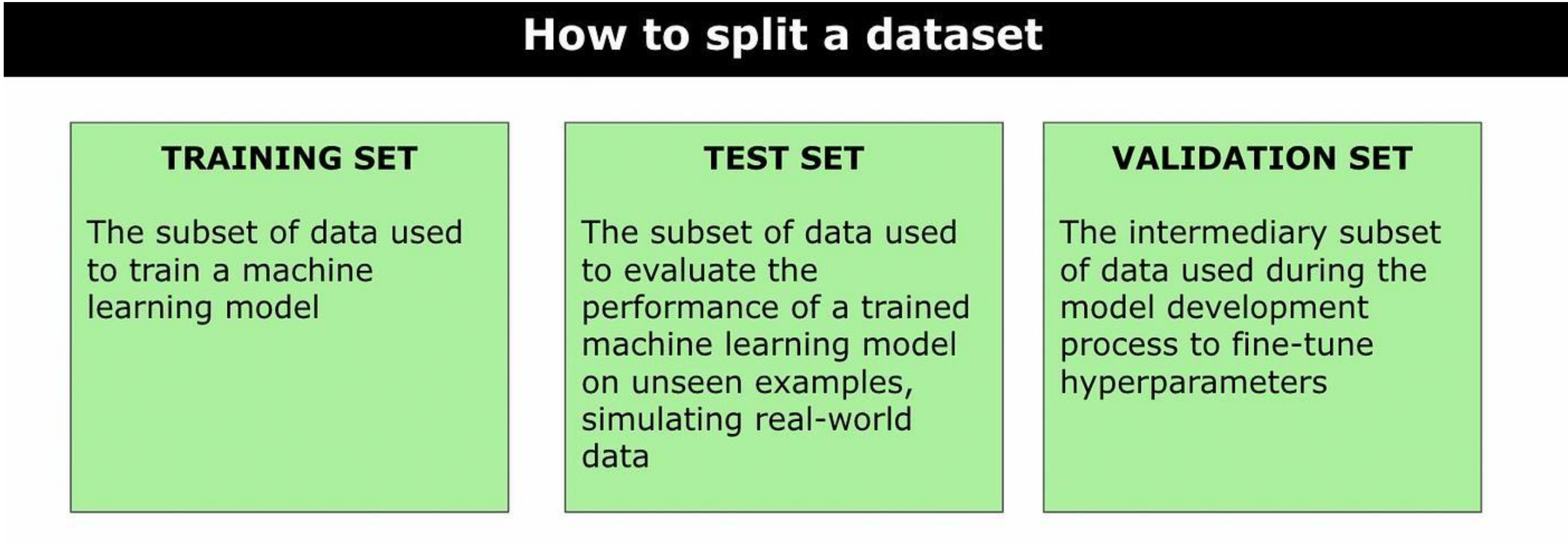
a) Age and Blood Pressure (0.65)

b) Sodium to Potassium and Drug (0.59)

2. Strongest negative correlation:

a)Cholesterol and Drug (-0.43)

Data Splitting, training and Testing



How to split a dataset

TRAINING SET

The subset of data used to train a machine learning model

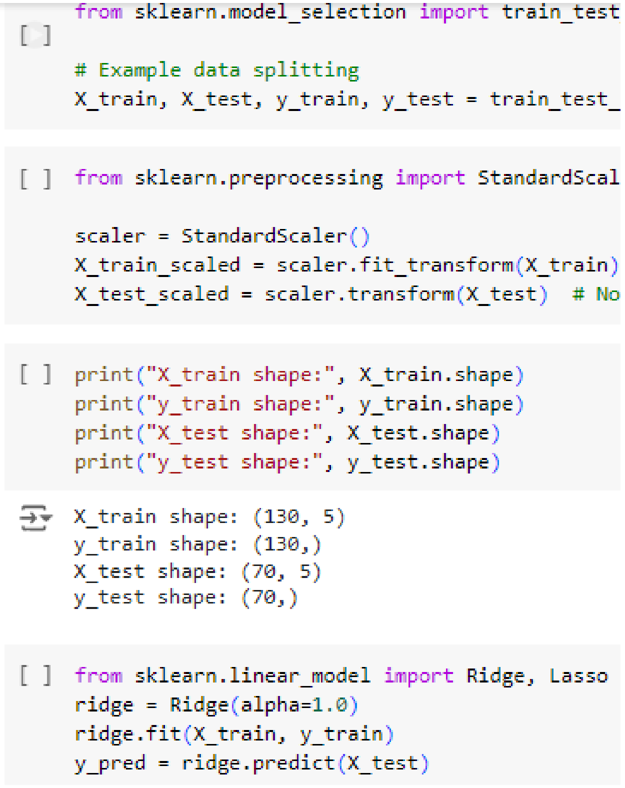
TEST SET

The subset of data used to evaluate the performance of a trained machine learning model on unseen examples, simulating real-world data

VALIDATION SET

The intermediary subset of data used during the model development process to fine-tune hyperparameters

Data Splitting, training and Testing



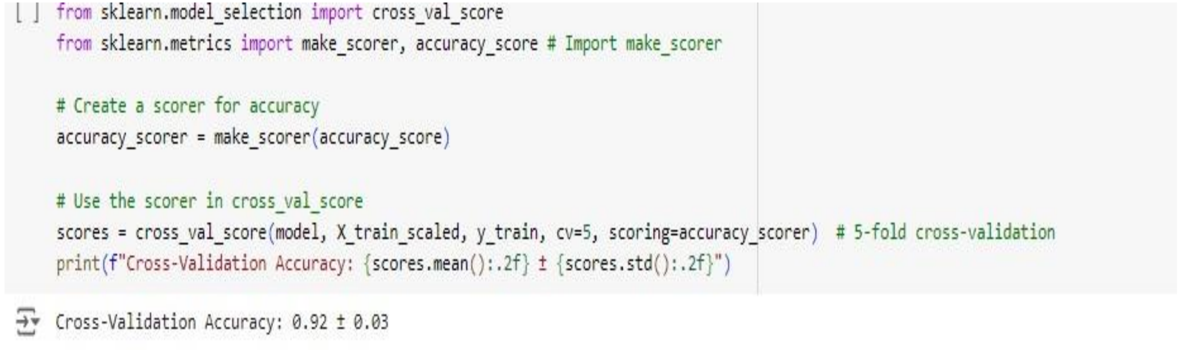
```
[ ]  from sklearn.model_selection import train_test

     # Example data splitting
     X_train, X_test, y_train, y_test = train_test_
```

```
[ ]  from sklearn.preprocessing import StandardScal

     scaler = StandardScaler()
     X_train_scaled = scaler.fit_transform(X_train)
     X_test_scaled = scaler.transform(X_test)  # No
```

```
[ ]  print("X_train shape:", X_train.shape)
     print("y_train shape:", y_train.shape)
     print("X_test shape:", X_test.shape)
     print("y_test shape:", y_test.shape)
```

```
X_train shape: (130, 5)
y_train shape: (130,)
X_test shape: (70, 5)
y_test shape: (70,)
```

```
[ ]  from sklearn.linear_model import Ridge, Lasso
     ridge = Ridge(alpha=1.0)
     ridge.fit(X_train, y_train)
     y_pred = ridge.predict(X_test)
```

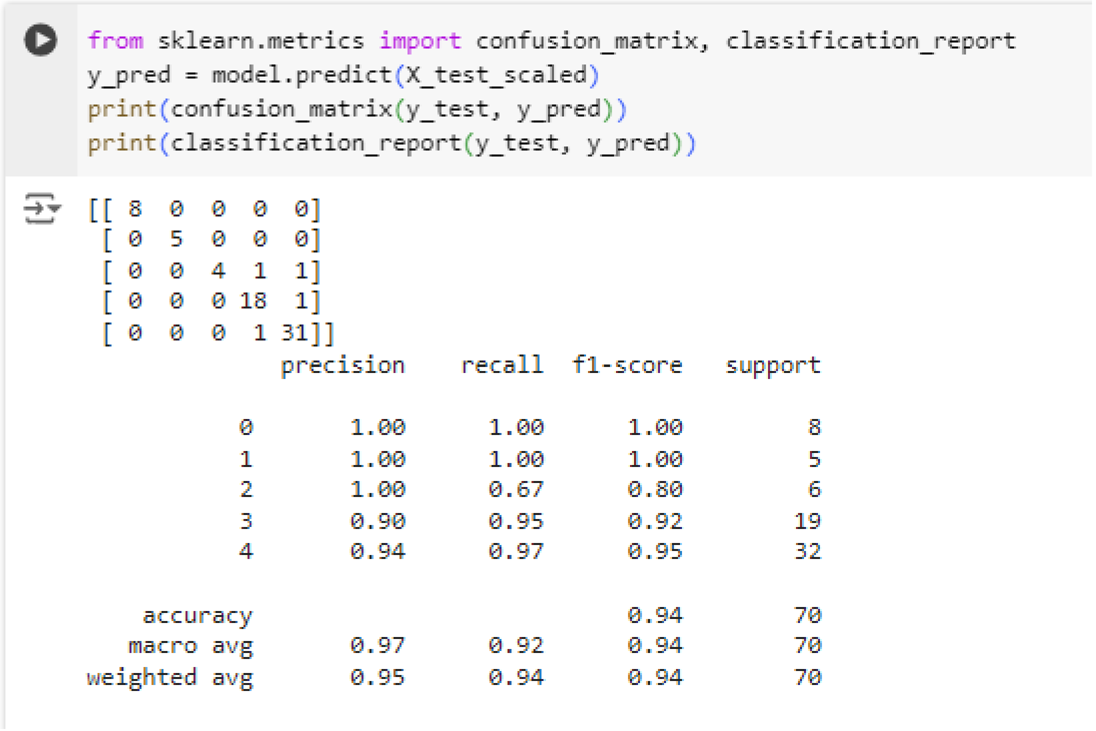


```
[ ]  from sklearn.model_selection import cross_val_score
     from sklearn.metrics import make_scorer, accuracy_score # Import make_scorer

     # Create a scorer for accuracy
     accuracy_scorer = make_scorer(accuracy_score)

     # Use the scorer in cross_val_score
     scores = cross_val_score(model, X_train_scaled, y_train, cv=5, scoring=accuracy_scorer)  # 5-fold cross-validation
     print(f"Cross-Validation Accuracy: {scores.mean():.2f} ± {scores.std():.2f}")
```

```
Cross-Validation Accuracy: 0.92 ± 0.03
```

The mean accuracy of 0.92 means that, on average, the model is correct 92% of the time across the 5 folds of cross-validation. The ±0.03 indicates that there is a small amount of variability in accuracy between different folds, suggesting that the model's performance is relatively stable across different subsets of the data. This is a strong result, reflecting that the model is likely to generalize well to new data.

Data Splitting, training and Testing



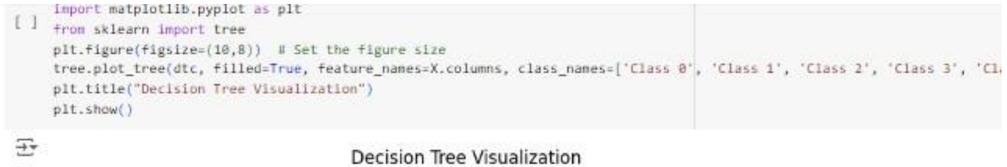
```
from sklearn.metrics import confusion_matrix, classification_report
y_pred = model.predict(X_test_scaled)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[ 8  0  0  0  0]
 [ 0  5  0  0  0]
 [ 0  0  4  1  1]
 [ 0  0  0 18  1]
 [ 0  0  0  1 31]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       1.00      1.00      1.00         5
           2       1.00      0.67      0.80         6
           3       0.90      0.95      0.92        19
           4       0.94      0.97      0.95        32

    accuracy                           0.94        70
   macro avg       0.97      0.92      0.94        70
weighted avg       0.95      0.94      0.94        70
```

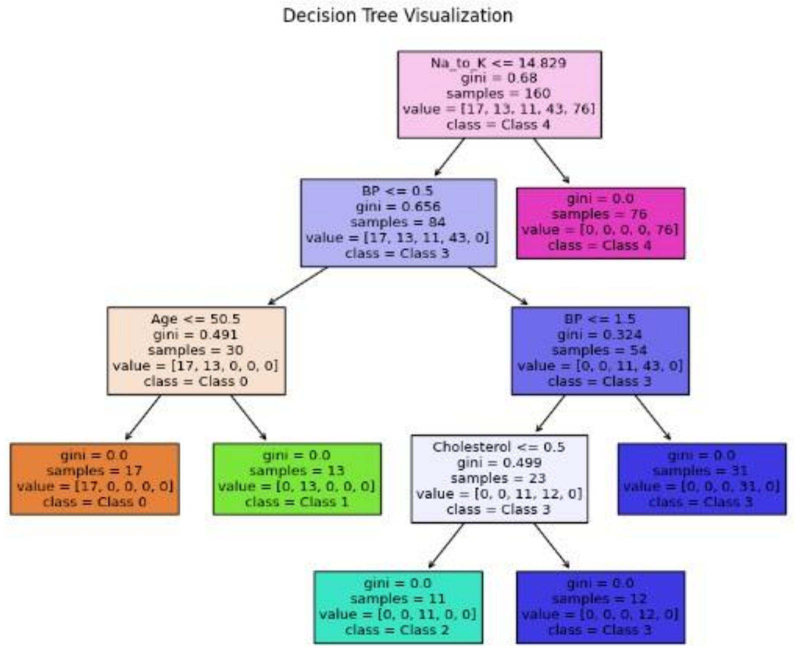
Confusion Matrix Analysis

- ➢ The model is consistently good at predicting the correct class with an accuracy of 94% and weighted average of 95%.
- ➢ It doesn't miss many true positives and doesn't make many mistakes in its predictions.

Training the Decision Tree



```
import matplotlib.pyplot as plt
from sklearn import tree
plt.figure(figsize=(10,8))  # Set the figure size
tree.plot_tree(dtc, filled=True, feature_names=X.columns, class_names=['Class 0', 'Class 1', 'Class 2', 'Class 3', 'Cl
plt.title("Decision Tree Visualization")
plt.show()
```


Decision Tree Visualization
Na_to_K <= 14.829 gini = 0.68 samples = 160 value = [17, 13, 11, 43, 76] class = Class 4
BP <= 0.5 gini = 0.656 samples = 84 value = [17, 13, 11, 43, 0] class = Class 3
gini = 0.0 samples = 76 value = [0, 0, 0, 0, 76] class = Class 4
Age <= 50.5 gini = 0.491 samples = 30 value = [17, 13, 0, 0, 0] class = Class 0
BP <= 1.5 gini = 0.324 samples = 54 value = [0, 0, 11, 43, 0] class = Class 3
gini = 0.0 samples = 17 value = [17, 0, 0, 0, 0] class = Class 0
gini = 0.0 samples = 13 value = [0, 13, 0, 0, 0] class = Class 1
Cholesterol <= 0.5 gini = 0.499 samples = 23 value = [0, 0, 11, 12, 0] class = Class 3
gini = 0.0 samples = 31 value = [0, 0, 0, 31, 0] class = Class 3
gini = 0.0 samples = 11 value = [0, 0, 11, 0, 0] class = Class 2
gini = 0.0 samples = 12 value = [0, 0, 0, 12, 0] class = Class 3

Decision Tree Analysis

- ❖ Na_to_K <= 14.83, BP <= 0.50, Age <= 50.50: This path results in class: 0.
- ❖ Na_to_K <= 14.83, BP <= 0.50, Age > 50.50: This path results in class: 1.

- ❖ Na_to_K <= 14.83, BP > 0.50, BP <= 1.50, Cholesterol <= 0.50: This path results in class: 2.
- ❖ Na_to_K <= 14.83, BP > 0.50, BP <= 1.50, Cholesterol > 0.50: This path results in class: 3.
- ❖ Na_to_K > 14.83: This path results in class: 4

Summary and Recommendations

- The precision being all at 95% , recall 94% and F1 score 94% suggests that the model is both highly accurate and reliable in predicting the classes.

Recommendation

a) Improvement Needed for Class 2: It struggles with Class2, missing items.

- Look into why the model misses many Class 2 items and make adjustments to improve it.
- Continue to check and refine the model to ensure it works well for all the classes.

References

- Gomez, P. (2021). *Drugs A, B, C, X, Y for Decision Trees.* https://www.kaggle.com/datasets/pablomgomez21/drugs-a-b-c-x-y-for-decision-trees
- Maina, N. (2024). Tableau Dashboard. https://public.tableau.com/views/DrugABCXYMachineLearningproject/Dashboard2?:language=en-US&publish=yes&:sid=&:redirect=auth&:display_count=n&:origin=viz_share_link
- Maina, N. (2024). Github Repo. https://github.com/Nmwangu2/Phase3_project.git

QUESTIONS