Patient-Centric Drug Selection using Machine Learning

Step into the realm of healthcare innovation, the patient-centric drug selector, in our project, "Patient-Centric Drug Selection using Machine Learning." This project offers an immersive journey into data analysis, model building, and evaluation using machine learning techniques to revolutionize patient care through accurate drug predictions.

The primary objective of this project is to leverage machine learning to predict the most suitable medication for patients based on vital factors like age, sex, blood pressure, and other critical attributes. Through hands-on experience in handling medical data and applying machine learning concepts, we will explore the realms of healthcare analytics and its potential to optimize patient-centric drug selection.

The dataset encompasses crucial patient attributes such as age, sex, blood pressure levels (BP), cholesterol levels, and the sodium-to-potassium ratio. Our focus lies in utilizing these features to predict the ideal drug type for patients, thereby enabling personalized and effective medical treatment decisions.

This venture isn't solely about predicting drug types; it's a transformative journey that seeks to enhance patient outcomes by offering tailored and precise drug predictions. The dataset provides an exceptional opportunity to delve into healthcare-focused machine learning techniques, aiming to revolutionize the delivery of healthcare services.

Part 1

1: Data Import for Patient-Centric Drug Selection In our project, "Patient-Centric Drug Selection using Machine Learning," this initial task involves importing a drug dataset ('drug_dataset.csv'). By loading this dataset, we lay the groundwork for our analysis and machine learning model development. This step is crucial as it provides the fundamental data needed for identifying patterns and features relevant to patient-centric drug selection.

In [12]:
```
import pandas as pd
df=pd.read_csv('/Users/faithedafetanureibeh/Downloads/drug_dataset.csv')
df
```

Out[12]:

	Age	Sex	BP	Cholesterol	Allergy	Na_to_K	Medication_Duration	Drug
0	23	F	HIGH	HIGH	Pollen	25.355	40	DrugY
1	47	M	LOW	HIGH	None	13.093	25	drugC
2	47	M	LOW	HIGH	Peanuts	10.114	25	drugC
3	28	F	NORMAL	HIGH	Shellfish	7.798	120	drugX
4	61	F	LOW	HIGH	Pollen	18.043	40	DrugY
...	...	...	...	...	...	...	...	...
234	35	M	NORMAL	NORMAL	None	7.845	40	drugX
235	47	M	LOW	NORMAL	None	33.542	25	DrugY
236	32	F	NORMAL	HIGH	None	7.477	25	drugX
237	70	F	NORMAL	HIGH	Pollen	20.489	120	DrugY
238	52	M	LOW	NORMAL	None	32.922	40	DrugY

239 rows × 8 columns

2: Assessing Null Values in Drug Dataset

In our project, "Patient-Centric Drug Selection using Machine Learning," this task involves assessing null values within the drug dataset ('drug_dataset.csv'). The 'sumofnull' calculation provides insight into the extent of missing values in the dataset. Identifying and understanding null values is critical for data preprocessing, ensuring that our machine learning model is built on complete and reliable information for accurate drug selection.

In [13]:
```
sumofnull=df.isnull().sum()
sumofnull
```

Out[13]:
```
Age                  0
Sex                  0
BP                   0
Cholesterol          0
Allergy              0
Na_to_K              0
Medication_Duration  0
Drug                 0
dtype: int64
```

3: Analyzing Data Types in Drug Dataset¶

In our project, "Patient-Centric Drug Selection using Machine Learning," this task involves examining the data types present in the drug dataset ('drug_dataset.csv'). The 'dtype' calculation provides information about the types of data present in each column. Understanding data types is essential for feature engineering and data preprocessing, enabling us to appropriately handle and transform data for the machine learning model's effective training and drug selection process.

In [14]:
```
dtype=df.dtypes
dtype
```

Out[14]:
```
Age                    int64
Sex                   object
BP                    object
Cholesterol           object
Allergy               object
Na_to_K              float64
Medication_Duration    int64
Drug                  object
dtype: object
```

4: Descriptive Statistics of Drug Dataset

In our project, "Patient-Centric Drug Selection using Machine Learning," this task focuses on generating descriptive statistics for the drug dataset ('drug_dataset.csv'). The 'describe' function provides statistical insights such as mean, standard deviation, minimum, maximum, and quartile information for numerical columns. Understanding these statistical measures is crucial for gaining initial insights into the dataset's characteristics, aiding in feature selection and model building for accurate patient-centric drug selection using machine learning techniques.

In [15]:
```
describe = df.describe()
describe
```

Out[15]:

	Age	Na_to_K	Medication_Duration
count	239.000000	239.000000	239.000000
mean	43.669456	16.263845	47.531381
std	16.612854	7.248121	32.786777
min	15.000000	6.269000	25.000000
25%	30.500000	10.571000	25.000000
50%	43.000000	13.967000	40.000000
75%	58.000000	20.251000	40.000000
max	74.000000	38.247000	120.000000

5: Analyzing Target Variable Distribution¶

In our project, "Patient-Centric Drug Selection using Machine Learning," this task involves analyzing the distribution of the target variable, 'Drug,' within the dataset. The 'target_count' calculation provides a count of occurrences for each drug category. Understanding the distribution of the target variable is crucial for ensuring the dataset's balance and assessing potential biases, which significantly influences the machine learning model's training and predictive accuracy in drug selection for patients.

In [16]:
```
target_count=df['Drug'].value_counts()
target_count
```

Out[16]:
```
DrugY    111
drugX     61
drugA     29
drugB     20
drugC     18
Name: Drug, dtype: int64
```

Part 2

1: Identifying Duplicate Records

In our project, "Patient-Centric Drug Selection using Machine Learning," this task involves identifying and quantifying duplicate records within the dataset. The 'duplicates' count helps determine the presence of repeated entries. Handling duplicate records ensures the dataset's cleanliness and avoids potential

biases or inaccuracies during model training for patient-centric drug selection using machine learning algorithms.

In [17]:
```
duplicates=df.duplicated().sum()
duplicates
```

Out[17]: 4

2: Removing Duplicate Records

In our project, "Patient-Centric Drug Selection using Machine Learning," this task involves the removal of duplicate records from the dataset. By executing the code that eliminates duplicate entries, we ensure data cleanliness and accuracy in our subsequent analysis and model development. Removing duplicates is crucial to prevent biases and distortions that duplicated data might introduce during machine learning model training for patient-centric drug selection.

In [18]:
```
df.drop_duplicates(inplace=True)
```

3: Feature Selection in Drug Dataset

In our project, "Patient-Centric Drug Selection using Machine Learning," this task involves feature selection within the dataset. The code removes the 'Allergy' and 'Medication_Duration' columns from the data, indicating a focus on specific features for model development. Feature selection is crucial for enhancing model efficiency by using the most relevant and impactful features for patient-centric drug selection through machine learning algorithms.

In [19]:
```
data = df.copy()
data = data.drop(['Allergy','Medication_Duration'], axis=1)

data
```

Out[19]:

	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
...	...	...	...	...	...	...
233	66	M	HIGH	HIGH	16.347	DrugY
234	35	M	NORMAL	NORMAL	7.845	drugX
236	32	F	NORMAL	HIGH	7.477	drugX
237	70	F	NORMAL	HIGH	20.489	DrugY
238	52	M	LOW	NORMAL	32.922	DrugY

235 rows × 6 columns

4: Age Group Categorization and Feature Engineering

In our project, "Patient-Centric Drug Selection using Machine Learning," this task involves categorizing age into age groups and subsequent feature engineering. The 'get_age_group' function categorizes ages into specific groups, and then the 'AgeGroup' column is created based on these categories. Additionally, the original 'Age' column is dropped from the dataset. This feature engineering step simplifies age data, making it more conducive for machine learning model training in the context of patient-centric drug selection.

In [20]:

```
def get_age_group(age):
    if age <= 30:
        return '0-30'
    elif age <= 40:
        return '30-40'
    elif age <= 50:
        return '40-50'
    elif age <= 60:
        return '50-60'
    else:
        return '60+'

data['AgeGroup'] = data['Age'].apply(get_age_group)

data = data.drop(['Age'], axis=1)

data
```

Out[20]:

	Sex	BP	Cholesterol	Na_to_K	Drug	AgeGroup
0	F	HIGH	HIGH	25.355	DrugY	0-30
1	M	LOW	HIGH	13.093	drugC	40-50
2	M	LOW	HIGH	10.114	drugC	40-50
3	F	NORMAL	HIGH	7.798	drugX	0-30
4	F	LOW	HIGH	18.043	DrugY	60+
...	...	...	...	...	...	...
233	M	HIGH	HIGH	16.347	DrugY	60+
234	M	NORMAL	NORMAL	7.845	drugX	30-40
236	F	NORMAL	HIGH	7.477	drugX	30-40
237	F	NORMAL	HIGH	20.489	DrugY	60+
238	M	LOW	NORMAL	32.922	DrugY	50-60

235 rows × 6 columns

5: Sodium-to-Potassium Ratio Grouping and Feature Engineering

In our project, "Patient-Centric Drug Selection using Machine Learning," this task involves categorizing the sodium-to-potassium ratio ('Na_to_K') into specific groups and performing feature engineering. The 'get_na_to_k_group' function categorizes values into distinct groups, and a new column, 'Na_to_K_groups,' is created based on these groupings. Subsequently, the original 'Na_to_K' column is dropped from the dataset. This feature engineering step simplifies the sodium-to-potassium ratio data, making it more suitable for machine learning model training in the context of patient-centric drug selection.

In [21]:
```
def get_na_to_k_group(value):
  if value <= 10:
      return '5-10'
  elif value <= 15:
      return '10-15'
  elif value <= 20:
      return '15-20'
  elif value <= 25:
      return '20-25'
  elif value <= 30:
      return '25-30'
  else:
      return '30+'

data['Na_to_K_groups'] = data['Na_to_K'].apply(get_na_to_k_group)
```

```
data = data.drop(['Na_to_K'], axis=1)
data
```

Out[21]:

	Sex	BP	Cholesterol	Drug	AgeGroup	Na_to_K_groups
0	F	HIGH	HIGH	DrugY	0-30	25-30
1	M	LOW	HIGH	drugC	40-50	10-15
2	M	LOW	HIGH	drugC	40-50	10-15
3	F	NORMAL	HIGH	drugX	0-30	5-10
4	F	LOW	HIGH	DrugY	60+	15-20
...	...	...	...	...	...	...
233	M	HIGH	HIGH	DrugY	60+	15-20
234	M	NORMAL	NORMAL	drugX	30-40	5-10
236	F	NORMAL	HIGH	drugX	30-40	5-10
237	F	NORMAL	HIGH	DrugY	60+	20-25
238	M	LOW	NORMAL	DrugY	50-60	30+

235 rows × 6 columns

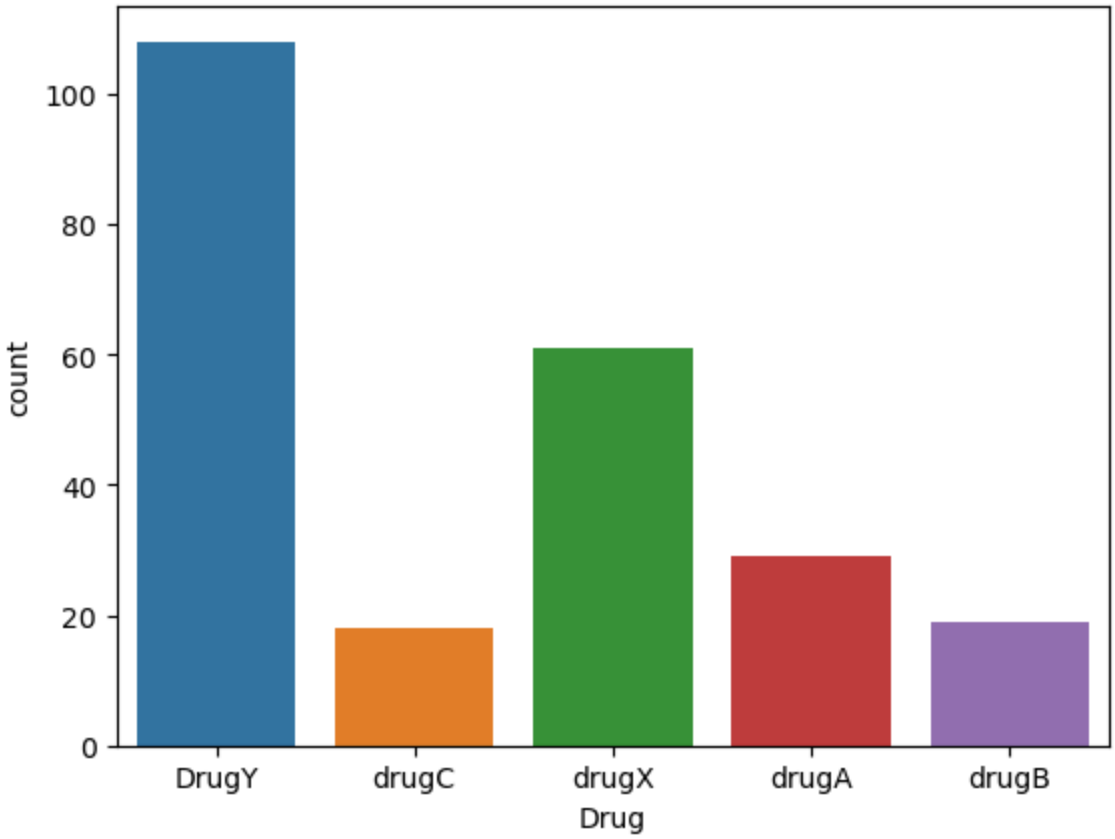
Part 3

1: Visualizing Drug Distribution

In our project, "Patient-Centric Drug Selection using Machine Learning," this task involves visualizing the distribution of drugs within the dataset. The 'sns.countplot' creates a count plot representing the frequency of each drug category. Visualizing drug distribution aids in understanding the imbalance or distribution of drugs in the dataset, which is essential for assessing potential biases and informing model training for patient-centric drug selection using machine learning techniques.

In [22]:

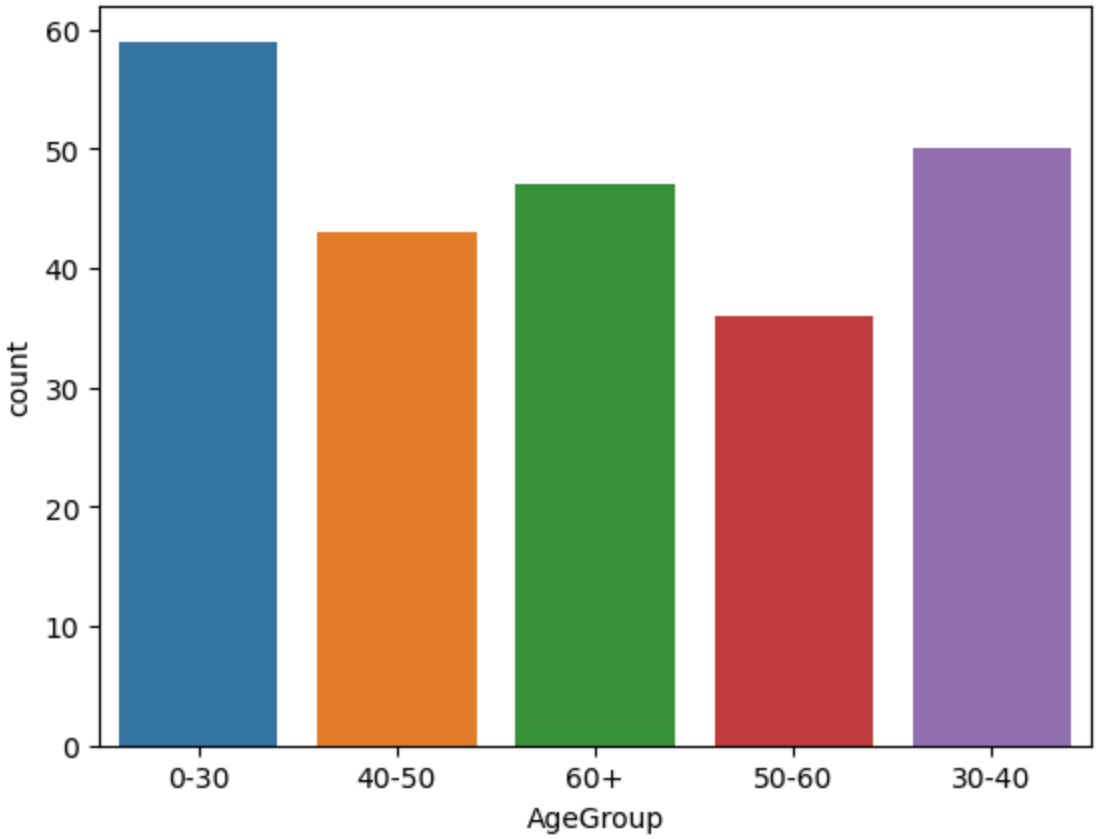
```
import seaborn as sns
import matplotlib.pyplot as plt

drug_ax = sns.countplot(x=data['Drug'])
```



2: Visualizing Age Group Distribution¶

In our project, "Patient-Centric Drug Selection using Machine Learning," this task involves visualizing the distribution of age groups within the dataset. The 'sns.countplot' generates a count plot illustrating the frequency of each age group. Visualizing age group distribution assists in understanding the distribution of age categories in the dataset, enabling insights into the representation of different age groups crucial for patient-centric drug selection analysis and model training.

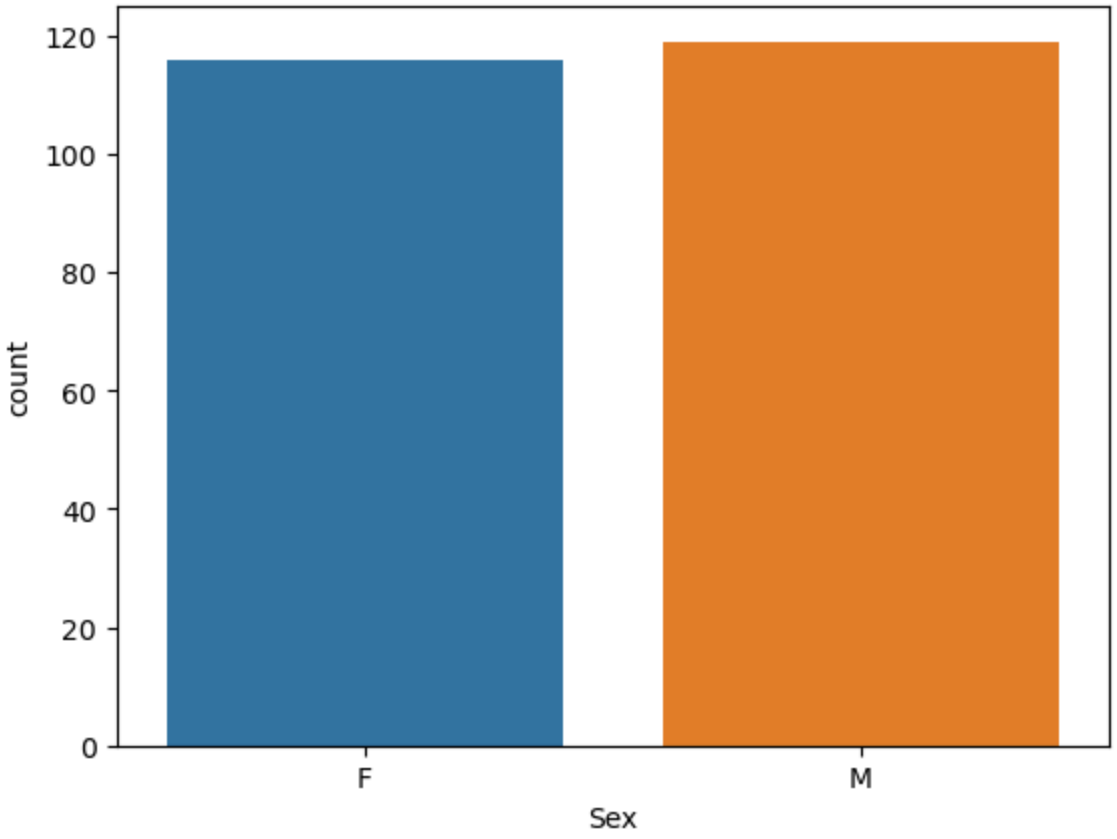
In [23]:
```
agegrp_ax = sns.countplot(x=data['AgeGroup'])
```



3: Visualizing Gender Distribution¶

In our project, "Patient-Centric Drug Selection using Machine Learning," this task involves visualizing the distribution of gender ('Sex') within the dataset. The 'sns.countplot' creates a count plot representing the frequency of each gender category. Visualizing gender distribution assists in understanding the representation of different genders in the dataset, which is essential for analyzing any gender-based patterns in patient-centric drug selection and model development.

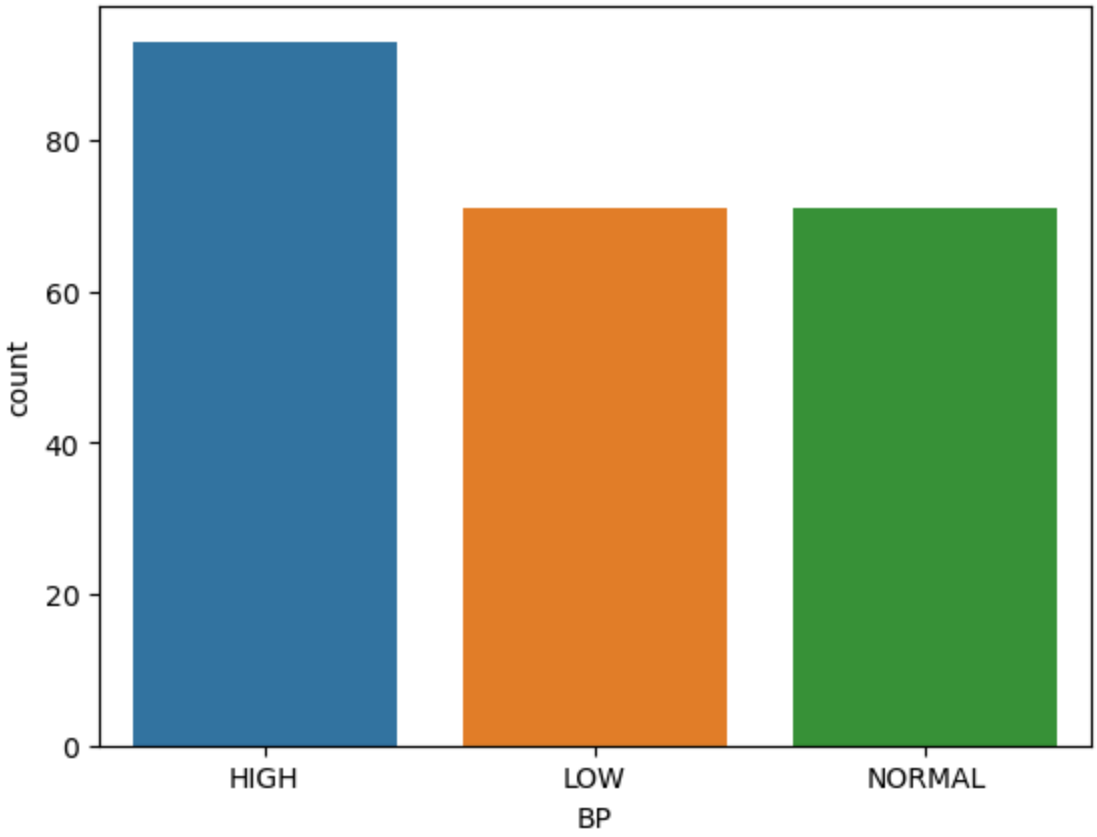
In [24]:
```
sex_ax = sns.countplot(x=data["Sex"])
```



4: Visualizing Blood Pressure Distribution

In our project, "Patient-Centric Drug Selection using Machine Learning," this task involves visualizing the distribution of blood pressure ('BP') categories within the dataset. The 'sns.countplot' generates a count plot illustrating the frequency of each blood pressure category. Visualizing blood pressure distribution assists in understanding the prevalence of different blood pressure levels among patients, a critical factor for drug selection analysis and modeling based on blood pressure levels.

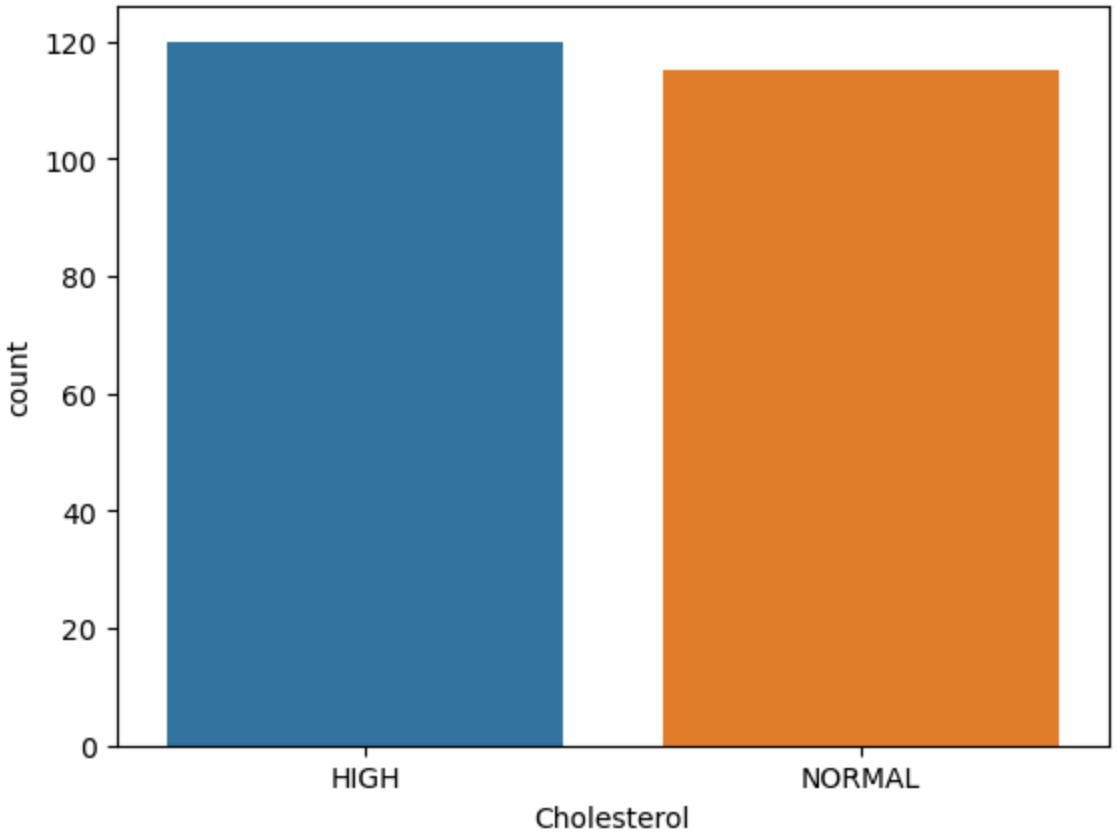
In [25]:
```
bp_ax = sns.countplot(x=data["BP"])
```



5: Visualizing Cholesterol Level Distribution¶

In our project, "Patient-Centric Drug Selection using Machine Learning," this task involves visualizing the distribution of cholesterol levels within the dataset. The 'sns.countplot' creates a count plot illustrating the frequency of each cholesterol level category. Visualizing cholesterol level distribution aids in understanding the distribution of cholesterol levels among patients, which is crucial information for drug selection analysis and modeling based on cholesterol levels.

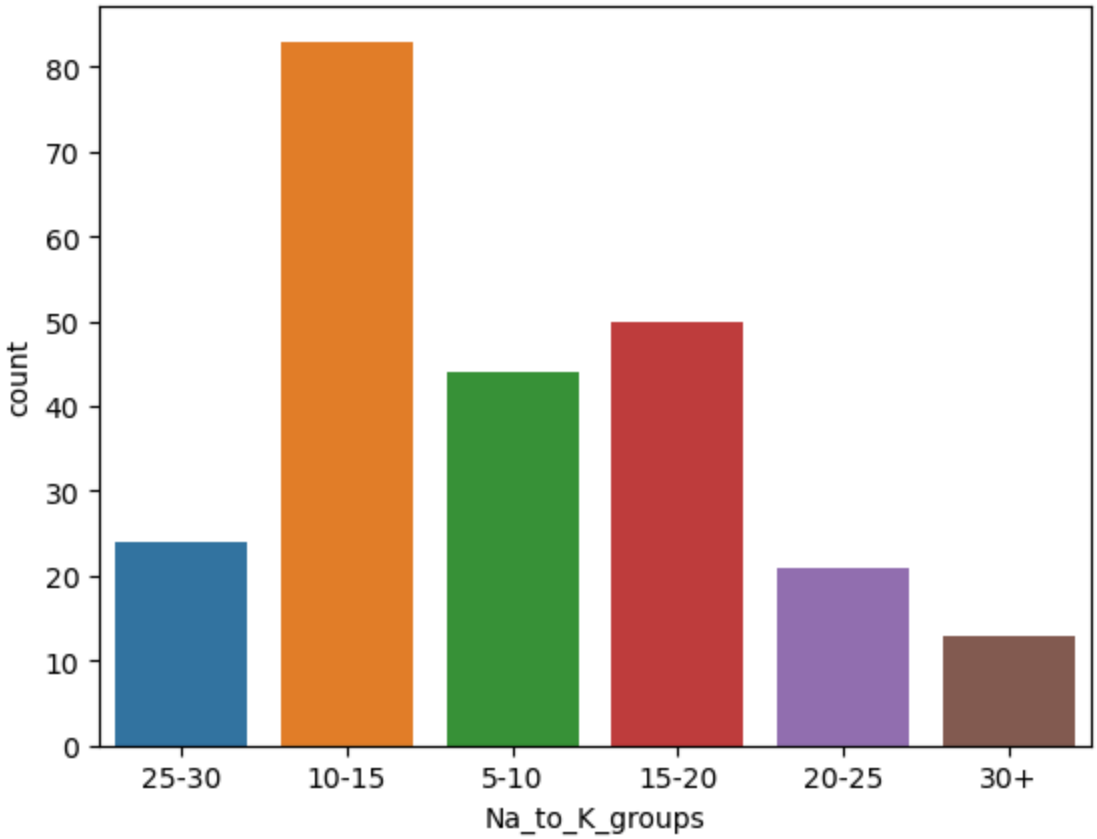
In [26]:
```
cholestrol_ax = sns.countplot(x=data["Cholesterol"])
```



6: Visualizing Sodium-to-Potassium Ratio Groups¶

In our project, "Patient-Centric Drug Selection using Machine Learning," this task involves visualizing the distribution of sodium-to-potassium ratio groups ('Na_to_K_groups') within the dataset. The 'sns.countplot' generates a count plot illustrating the frequency of each sodium-to-potassium ratio group category. Visualizing these ratio groups assists in understanding their distribution among patients, aiding drug selection analysis and modeling based on different sodium-to-potassium ratio categories.

In [27]:
```
na_ax = sns.countplot(x=data["Na_to_K_groups"])
```



Part 4

1: Data Splitting for Model Training and Evaluation

In our project, "Patient-Centric Drug Selection using Machine Learning," this task involves splitting the dataset into feature variables (X) and the target variable (y). Following this, the data is divided into training and testing sets using the train_test_split function from a machine learning library, likely scikit-learn. This process allocates 80% of the data for training (x_train, y_train) and 20% for testing (x_test, y_test) machine learning models that predict the 'Drug' variable based on other features. This step is critical for model development and evaluation in drug selection for patients.

In [28]:
```
from sklearn.model_selection import train_test_split

X = data.drop(columns='Drug')
y = data['Drug']

x_train, x_test, y_train, y_test = train_test_split(X, y, train_size=0.8, test_

x_train
```

Out[28]:

	Sex	BP	Cholesterol	AgeGroup	Na_to_K_groups
217	F	HIGH	NORMAL	0-30	15-20
150	M	HIGH	NORMAL	40-50	5-10
34	M	NORMAL	HIGH	50-60	10-15
223	F	HIGH	NORMAL	30-40	10-15
82	F	LOW	HIGH	30-40	5-10
...	...	...	...	...	...
129	F	NORMAL	HIGH	30-40	5-10
197	M	NORMAL	HIGH	50-60	5-10
183	F	HIGH	NORMAL	30-40	15-20
174	M	HIGH	NORMAL	40-50	10-15
122	M	NORMAL	HIGH	30-40	20-25

188 rows × 5 columns

2: Data Preprocessing and Handling Class Imbalance¶

In our project, "Patient-Centric Drug Selection using Machine Learning," this task involves converting categorical variables into numerical format using one-hot encoding via pd.get_dummies(). Additionally, the dataset is converted to integers using .astype(int). Moreover, the Synthetic Minority Over-sampling Technique (SMOTE) is applied to the training set (x_train, y_train) to address class imbalance by generating synthetic samples for the minority class. This preprocessing step ensures balanced class representation in the training data for the machine learning models used in drug selection for patients.

In [29]:
```
from imblearn.over_sampling import SMOTE

x_train = pd.get_dummies(pd.DataFrame(x_train))
x_test = pd.get_dummies(pd.DataFrame(x_test))
x_train = x_train.astype(int)
x_test = x_test.astype(int)

x_train, y_train = SMOTE().fit_resample(x_train, y_train)

x_train
```

Out[29]:

	Sex_F	Sex_M	BP_HIGH	BP_LOW	BP_NORMAL	Cholesterol_HIGH	Cholesterol_NORMAL
0	1	0	1	0	0	0	1
1	0	1	1	0	0	0	1
2	0	1	0	0	1	1	0
3	1	0	1	0	0	0	1
4	1	0	0	1	0	1	0
...	...	...	...	...	...	...	...
420	1	0	0	1	0	0	1
421	1	0	0	1	0	0	1
422	0	1	0	0	1	0	1
423	1	0	0	1	0	0	1
424	1	0	0	0	1	1	0

425 rows × 18 columns

3: Model Training and Evaluation¶

In our project, "Patient-Centric Drug Selection using Machine Learning," this task involves the implementation of a Logistic Regression model using the x_train and y_train datasets. After training the model, predictions (Y_pred) are made using the x_test dataset. Subsequently, evaluation metrics including accuracy, precision, recall, and F1 score are computed using these predictions and y_test. These metrics help assess the model's performance in predicting drug selection for patients based on various features derived from the dataset.

In [35]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_s


model = LogisticRegression()

model.fit(x_train, y_train)

Y_pred = model.predict(x_test)

accuracy = accuracy_score(y_test, Y_pred)


precision = precision_score(y_test, Y_pred, average='weighted')


recall = recall_score(y_test, Y_pred, average='weighted')
```

```
f1 = f1_score(y_test, Y_pred, average='weighted')

accuracy,precision,recall,f1
```

Out[35]:
```
(1.0, 1.0, 1.0, 1.0)
```

4: Model Training and Evaluation with Naive Bayes Classifier¶

In our project, "Patient-Centric Drug Selection using Machine Learning," this task involves implementing a Gaussian Naive Bayes classifier using the x_train and y_train datasets. After training the model, predictions (Y_pred) are made using the x_test dataset. Subsequently, evaluation metrics including accuracy, precision, recall, and F1 score are computed using these predictions and y_test. These metrics help assess the performance of the Naive Bayes classifier in predicting drug selection for patients based on various features derived from the dataset, providing an alternative model for comparison with the Logistic Regression model.

In [34]:
```
from sklearn.naive_bayes import GaussianNB

model = GaussianNB()

model.fit(x_train, y_train)

Y_pred = model.predict(x_test)

accuracy = accuracy_score(y_test, Y_pred)


precision = precision_score(y_test, Y_pred, average='weighted')


recall = recall_score(y_test, Y_pred, average='weighted')

f1 = f1_score(y_test, Y_pred, average='weighted')

accuracy,precision,recall,f1
```

Out[34]:
```
(1.0, 1.0, 1.0, 1.0)
```

5: Model Training and Evaluation with Decision Tree Classifier¶

In our project, "Patient-Centric Drug Selection using Machine Learning," this task involves implementing a Decision Tree

classifier using the x_train and y_train datasets. After training the model, predictions (Y_pred) are made using the x_test dataset. Subsequently, evaluation metrics including accuracy, precision, recall, and F1 score are computed using these predictions and y_test. These metrics help assess the performance of the Decision Tree classifier in predicting drug selection for patients based on various features derived from the dataset, providing an additional model for comparison with previous models.

In [32]:
```
from sklearn.tree import DecisionTreeClassifier

# Create a Decision Tree classifier
model = DecisionTreeClassifier()

model.fit(x_train, y_train)

Y_pred = model.predict(x_test)

accuracy = accuracy_score(y_test, Y_pred)


precision = precision_score(y_test, Y_pred, average='weighted')


recall = recall_score(y_test, Y_pred, average='weighted')

f1 = f1_score(y_test, Y_pred, average='weighted')

accuracy,precision,recall,f1
```

Out[32]:
```
(0.9361702127659575, 0.9574468085106383, 0.9361702127659575, 0.94229529335912
3)
```

6: Model Training and Evaluation with Random Forest Classifier¶

In our project, "Patient-Centric Drug Selection using Machine Learning," this task involves implementing a Random Forest classifier using the x_train and y_train datasets. The model is configured with 100 trees (num_trees = 100) and a maximum of 3 features (max_features = 3). After training the model, predictions (Y_pred) are made using the x_test dataset. Subsequently, evaluation metrics including accuracy, precision, recall, and F1 score are computed using these predictions and y_test. These metrics help assess the performance of the Random Forest classifier in predicting drug selection for patients based on various features derived from the dataset, offering a more complex ensemble model for comparison with previous models.

In [33]:
```
from sklearn.ensemble import RandomForestClassifier

# Define (num_trees =100) and (max_features=3)
num_trees = 100
max_features = 3

# Create a Random Forest classifier with the specified number of trees and max
model = RandomForestClassifier(n_estimators=num_trees, max_features=max_featur

model.fit(x_train, y_train)

Y_pred = model.predict(x_test)

accuracy = accuracy_score(y_test, Y_pred)


precision = precision_score(y_test, Y_pred, average='weighted')


recall = recall_score(y_test, Y_pred, average='weighted')

f1 = f1_score(y_test, Y_pred, average='weighted')

accuracy,precision,recall,f1
```

Out[33]:
```
(0.9787234042553191,
 0.9858156028368793,
 0.9787234042553191,
 0.9796690307328605)
```

7: Making Prediction with Best Model¶

In this task for "Patient-Centric Drug Selection using Machine Learning," the aim is to employ the model with the best evaluation metrics from the previous experiments for predicting the outcome on a sample data point. The sample data, structured as sample_data, represents the features for which the drug prediction needs to be made. By utilizing the identified best-performing model, the code executes a prediction based on the provided sample data, offering insights into drug selection according to the model's learned patterns and associations

In [40]:
```
sample_data = pd.DataFrame([[1,0,1,0,0,0,1,1,0,0,0,0,0,1,0,0,0,0]],
                    columns=['Sex_F', 'Sex_M', 'BP_HIGH', 'BP_LOW', 'BP_NORMAL
                             'AgeGroup_30-40', 'AgeGroup_40-50', 'AgeGroup_50-(
                             'Na_to_K_groups_20-25', 'Na_to_K_groups_25-30', 'N

best_model = LogisticRegression()

best_model.fit(x_train, y_train)

prediction =best_model.predict(sample_data)

prediction
```

Out[40]:
```
array(['DrugY'], dtype=object)
```

In [41]:
```
sample_data = pd.DataFrame([[1,0,1,0,0,0,1,1,0,0,0,0,0,1,0,0,0,0]],
                    columns=['Sex_F', 'Sex_M', 'BP_HIGH', 'BP_LOW', 'BP_NORMAL
                             'AgeGroup_30-40', 'AgeGroup_40-50', 'AgeGroup_50-(
                             'Na_to_K_groups_20-25', 'Na_to_K_groups_25-30', 'N

best_model = GaussianNB()

best_model.fit(x_train, y_train)

predictionn =best_model.predict(sample_data)

predictionn
```

Out[41]:
```
array(['DrugY'], dtype='<U5')
```

In [ ]:

In [ ]:

In [ ]: