Classifying drugs

We start by importing the data set and understanding the data set well .BP(Blood Pressure): .Represents the blood pressure category of individuals .Cholesterol: .Represent the cholesterol level category of individual .Na_to_K(Sodium to Potassium in the blood)

In [56]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import warnings
warnings.filterwarnings("ignore")
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import accuracy_score, confusion_matrix, classification
```

In [57]:
```
data = pd.read_csv("drug200.csv")
df = data.copy()
df.head()
```

Out[57]:

	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

Our goal is given someone's age, sex, blood pressure, cholesterol, Na_to_Km, to predict what type of drugs they use. Possible drugs are as follows:

In [58]:
```
df.head()
```

Out[58]:

	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

In [59]:
```
df["Drug"].unique()
```

Out[59]:
```
array(['DrugY', 'drugC', 'drugX', 'drugA', 'drugB'], dtype=object)
```

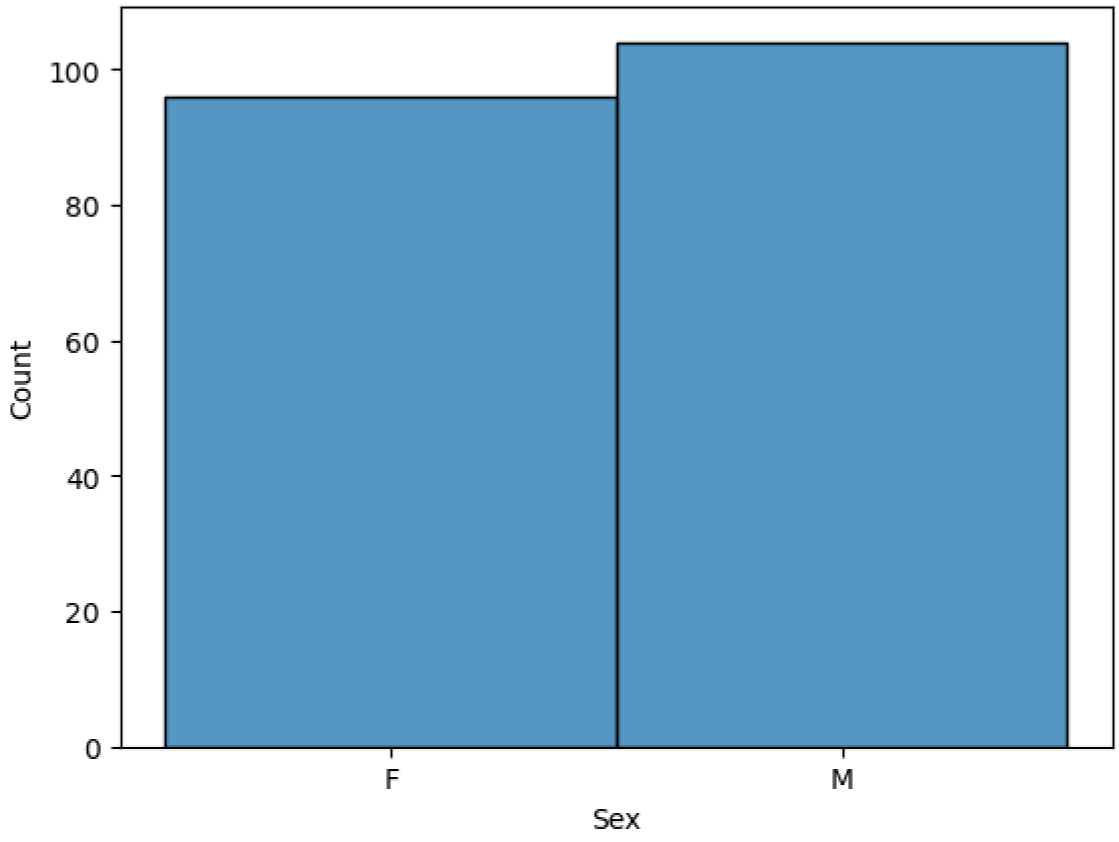
1. Study the data -The first plot tell us there aren't a big difference between males and females number -The second one tell us two things:

   1.The different drugs are consumed by almost the same amount of both genders

   2.Drug x and y are the most consumed

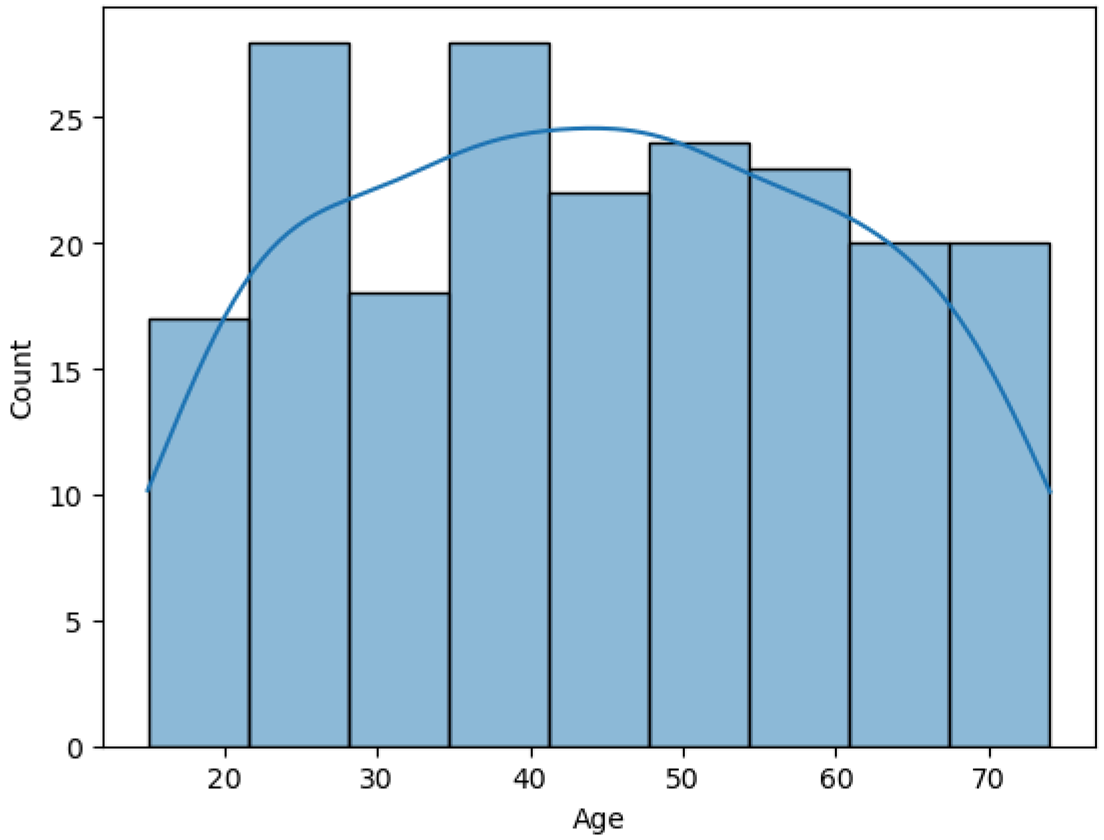
In [60]:
```
import warnings
warnings.filterwarnings("ignore")
sns.histplot(data=df, x="Sex")
```

Out[60]:
```
<Axes: xlabel='Sex', ylabel='Count'>
```



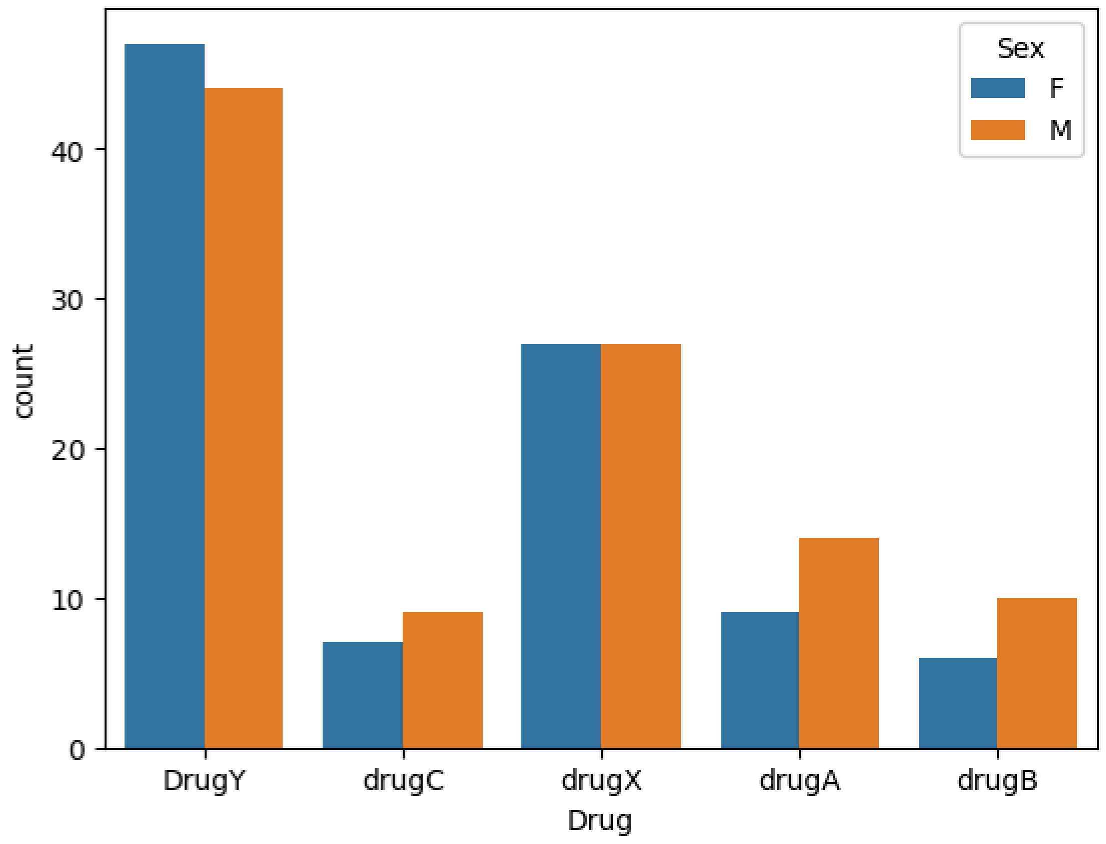
In [61]:
```
sns.histplot(data=df, x="Age", kde=True)
```

Out[61]:
```
<Axes: xlabel='Age', ylabel='Count'>
```



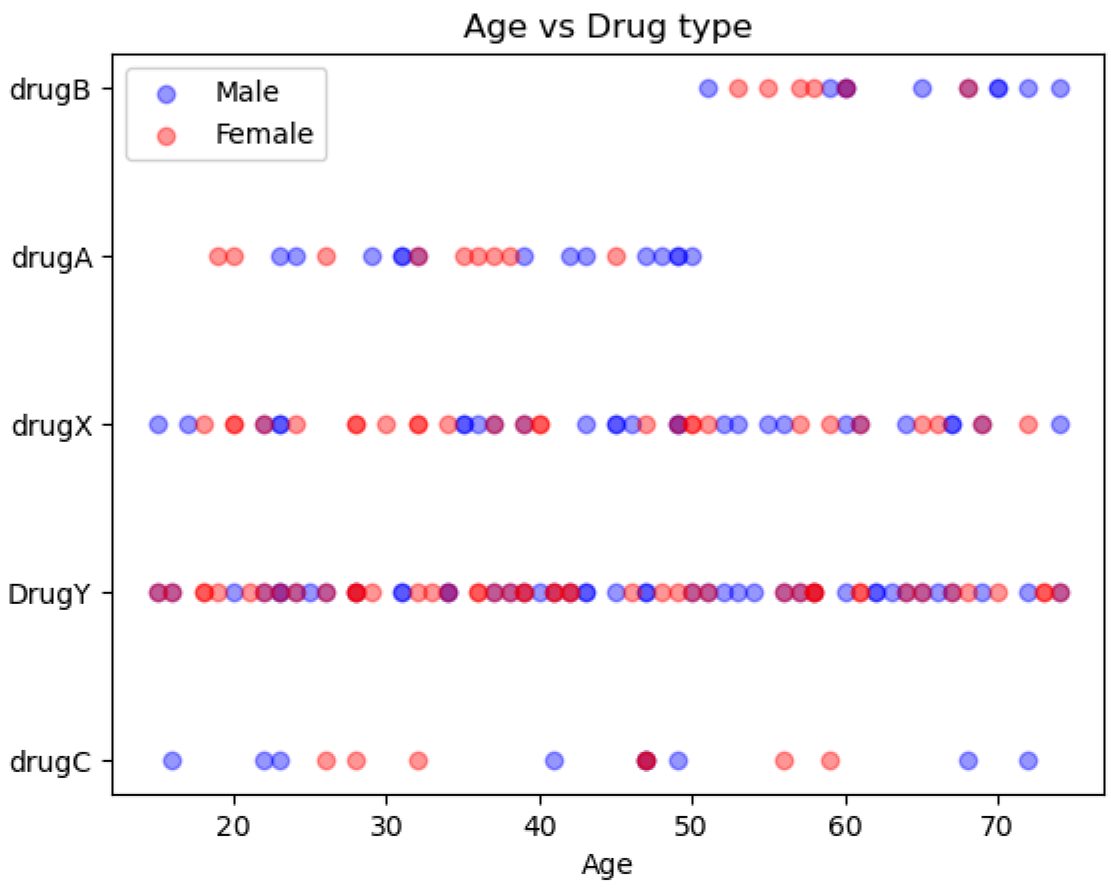
```
In [62]: sns.countplot(data=df, x="Drug", hue="Sex")
```

```
Out[62]: <Axes: xlabel='Drug', ylabel='count'>
```



In [63]:
```
male_drug = df[df["Sex"] == "M"]
female_drug = df[df["Sex"] == "F"]

plt.scatter(male_drug["Age"], male_drug["Drug"], alpha=0.4, color="blue", la
plt.scatter(female_drug["Age"], female_drug["Drug"], alpha=0.4, color="red",
plt.xlabel("Age")
plt.title("Age vs Drug type")
plt.legend()
```

Out[63]:
```
<matplotlib.legend.Legend at 0x11738c1afd0>
```



The next one tell us the relationship beetwen the age and the na-to-k, and if we see a pattern with the drugs that each person consume maybe we can conclude that the drug is what causes the level of na-to-k at a certain age.

.It's clearly that drugY altarate the level of na-to-k in any age

.Something similar for drugX cause no point pass the level of 15 na-to-k value

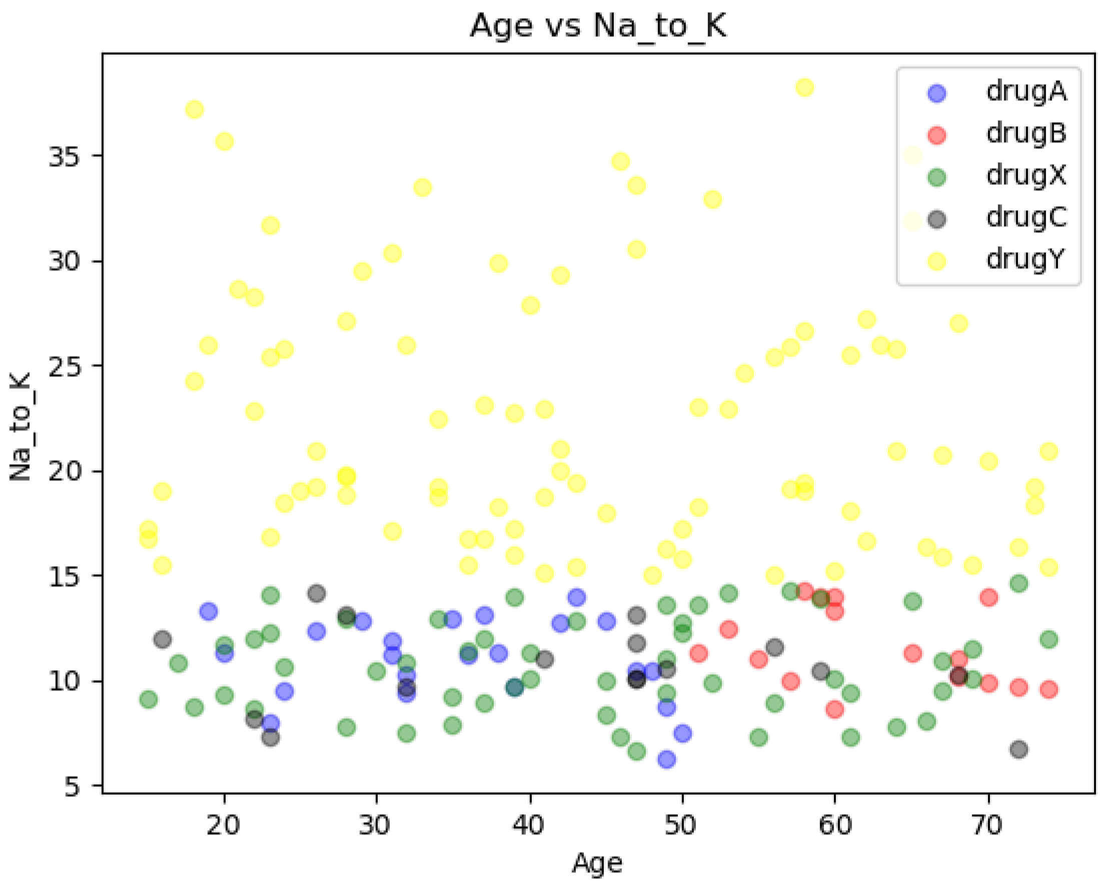
.drugB and drugA must be only for a specific range of age, because we clearly se a division around of 50 years

. Also of it, in both cases no one pass the level of 15 na-to-k value

.drugC have a more randomly patten, so we won't to conclude anything

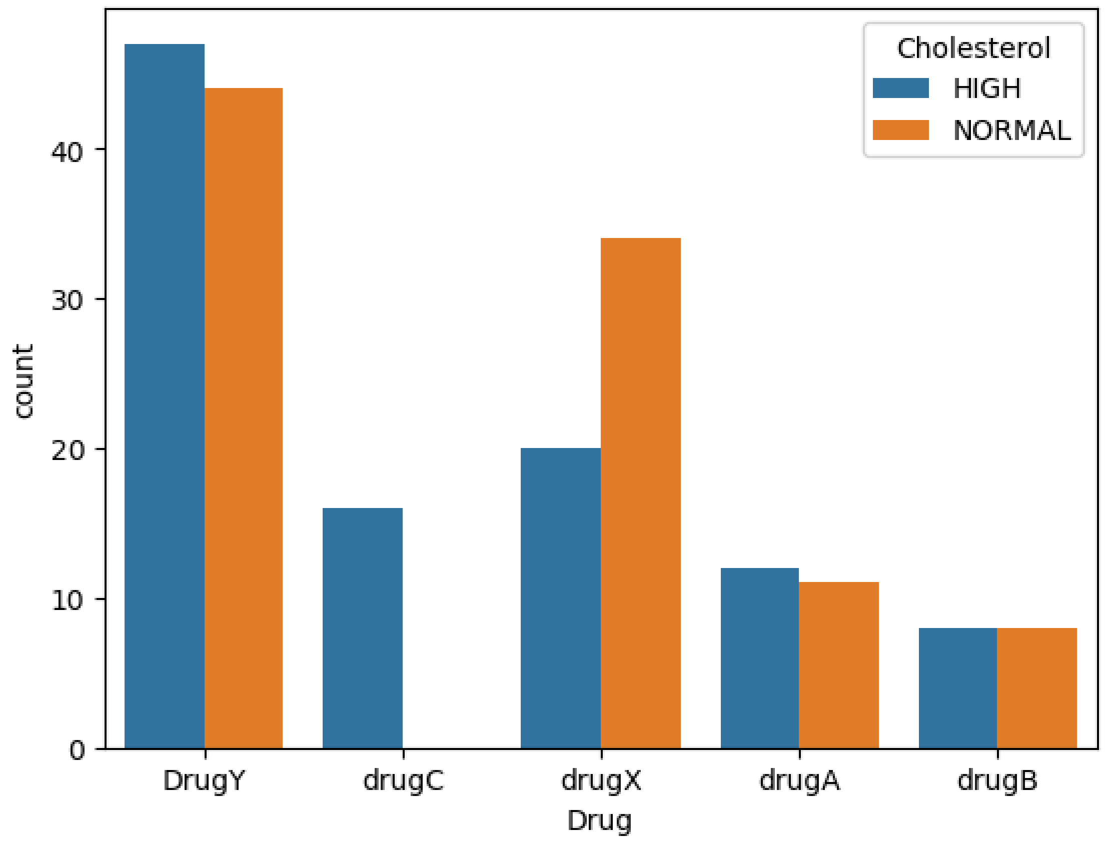
In [64]:
```
druga = df[df["Drug"] == "drugA"]
drugb = df[df["Drug"] == "drugB"]
drugx = df[df["Drug"] == "drugX"]
drugc = df[df["Drug"] == "drugC"]
drugy = df[df["Drug"] == "DrugY"]

plt.scatter(druga["Age"], druga["Na_to_K"], alpha=0.4, color="blue", label="
plt.scatter(drugb["Age"], drugb["Na_to_K"], alpha=0.4, color="red", label="d
plt.scatter(drugx["Age"], drugx["Na_to_K"], alpha=0.4, color="green", label=
plt.scatter(drugc["Age"], drugc["Na_to_K"], alpha=0.4, color="black", label=
plt.scatter(drugy["Age"], drugy["Na_to_K"], alpha=0.4, color="yellow", label
plt.xlabel("Age")
plt.ylabel("Na_to_K")
plt.title("Age vs Na_to_K")
plt.legend()
```

Out[64]:
```
<matplotlib.legend.Legend at 0x11738c53d50>
```



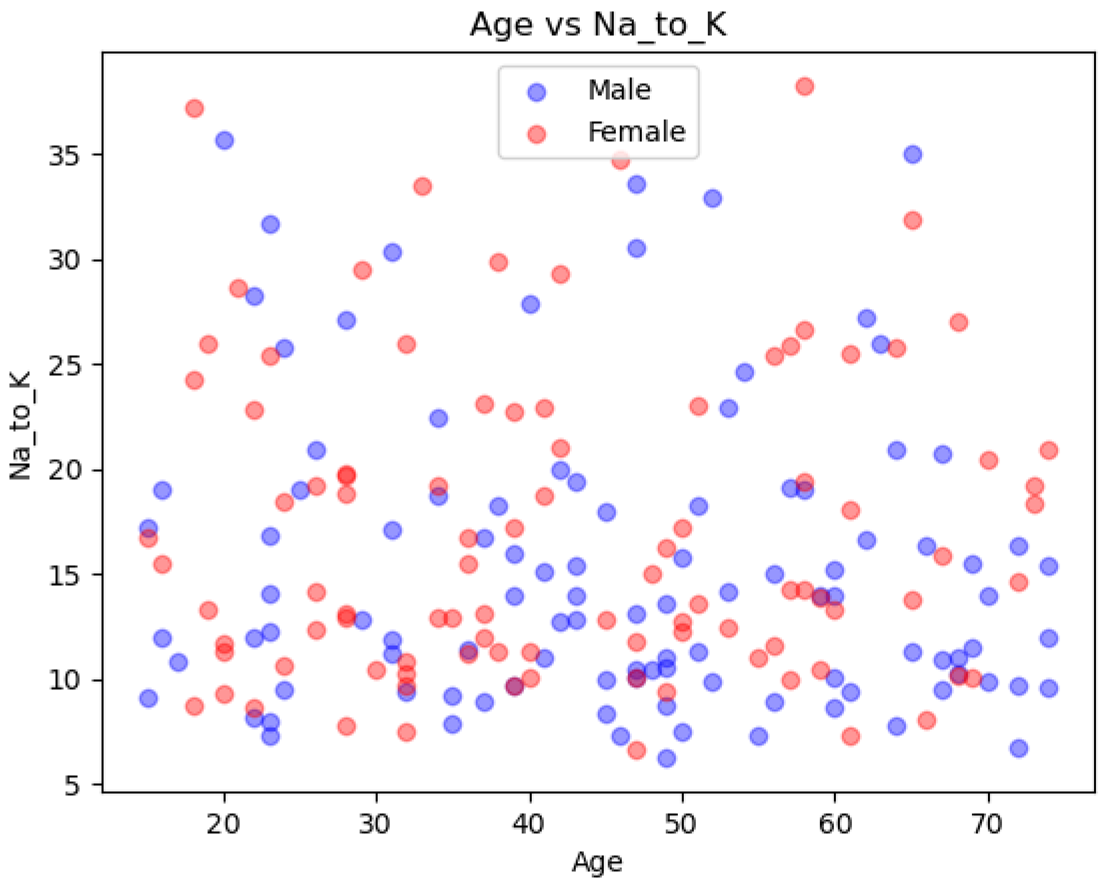
```
In [65]: sns.countplot(data=df, x="Drug", hue="Cholesterol")
```

```
Out[65]: <Axes: xlabel='Drug', ylabel='count'>
```



In [66]:
```
drug_male = df[df["Sex"] == "M"]
drug_female = df[df["Sex"] == "F"]

plt.scatter(drug_male["Age"], drug_male["Na_to_K"], alpha=0.4, color="blue",
plt.scatter(drug_female["Age"], drug_female["Na_to_K"], alpha=0.4, color="re
plt.xlabel("Age")
plt.ylabel("Na_to_K")
plt.title("Age vs Na_to_K")
plt.legend()
```

Out[66]:
```
<matplotlib.legend.Legend at 0x11738c8d3d0>
```



In my opinion the model will have a good accuracy because the groups are clearly differentiated in more than one feature

2. Convert categorical features into numerical and split the dataset

In [67]:
```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

cols2_encode = ["Sex", "BP", "Cholesterol", "Drug"]
encoders = {}
for col in cols2_encode:
    encoder = LabelEncoder()
    df[col] = encoder.fit_transform(df[col])
    encoders[col] = encoder

#encoders["Drug"].inverse_transform([1]) # to reverse the encoding

X = df.drop("Drug", axis=1)
y = df["Drug"].copy()

xtrain, xtest, ytrain, ytest = train_test_split(X, y, test_size=0.2, random_
```

3. Train the model In this case we will use the RandomForestClassifier and see how it works, So let's import it and then run the fit function with the traindataset

In [68]:
```
from sklearn.ensemble import RandomForestClassifier
forest_clf = RandomForestClassifier()
forest_clf.fit(xtrain, ytrain)
```

Out[68]:
```
RandomForestClassifier()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

1. Using cross_val_predict:

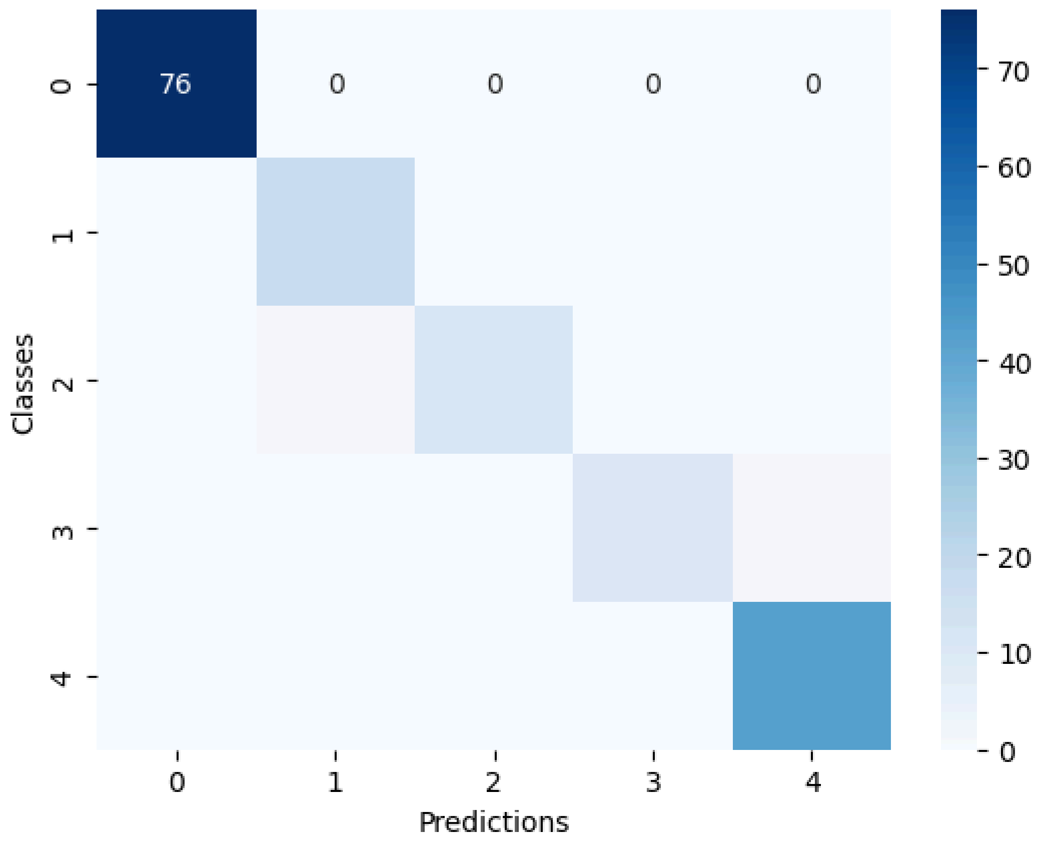
Is used to evaluate the performance of a machine It divides your dataset into several parts or "folds." For each fold, it trains the model on the other parts (e.g., 4 parts if you have 5 folds) and tests it on the remaining part After running through all the folds, it takes the average of the scores from each fold. This average score gives you a more reliable estimate of how well your model is expected to perform on new, unseen data

In [69]:
```
from sklearn.model_selection import cross_val_score
cross_val_score(forest_clf, xtrain, ytrain, cv=3, scoring="accuracy")
```

Out[69]:
```
array([0.98148148, 1.        , 0.96226415])
```

In [70]:
```
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_predict

ytrain_pred = cross_val_predict(forest_clf, xtrain, ytrain, cv=3)
conf_matrix = confusion_matrix(ytrain, ytrain_pred)
sns.heatmap(data=conf_matrix, annot=True, cmap="Blues")
plt.xlabel("Predictions")
plt.ylabel("Classes")
```

Out[70]:
```
Text(50.722222222222214, 0.5, 'Classes')
```



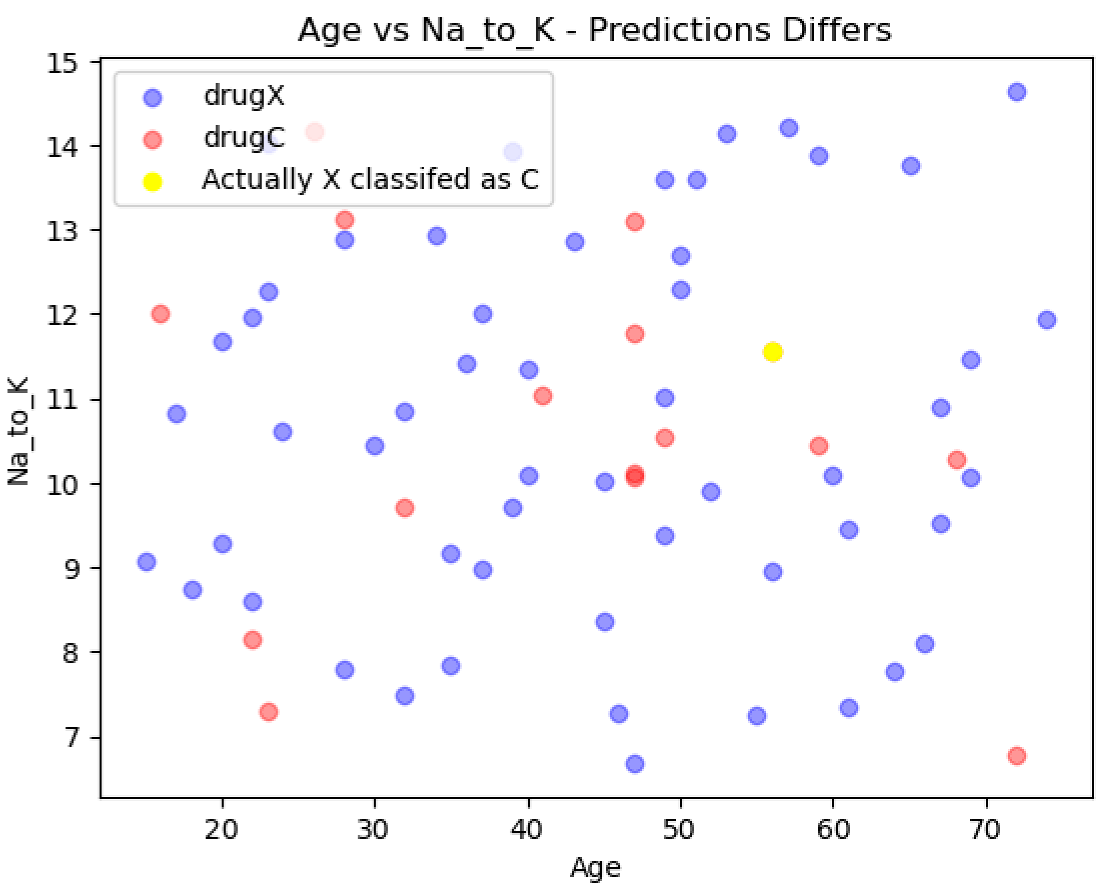
In [71]:
```
import numpy as np

class3 = encoders["Drug"].inverse_transform([3])[0]
class4 = encoders["Drug"].inverse_transform([4])[0]
print(f"class 3 is {class3}")
print(f"class 4 is {class4}")
```

```
class 3 is drugC
class 4 is drugX
```

. The Cholesterol histogram in male section hace almost the same value . In the Age vs Na_to_K and Age vs Drug type graphics we see that drugC and drugX cover the same range

In [73]:
```
misclassified_indices = np.where((ytrain_pred == 4) & (ytrain == 3))[0]
misclassified_xtrain = xtrain.iloc[misclassified_indices]

drugx = data[data["Drug"] == "drugX"]
drugc = data[data["Drug"] == "drugC"]

plt.scatter(drugx["Age"], drugx["Na_to_K"], alpha=0.4, color="blue", label="
plt.scatter(drugc["Age"], drugc["Na_to_K"], alpha=0.4, color="red", label="d
plt.scatter(misclassified_xtrain["Age"], misclassified_xtrain["Na_to_K"], al
plt.xlabel("Age")
plt.ylabel("Na_to_K")
plt.title("Age vs Na_to_K - Predictions Differs")
plt.legend()
```

Out[73]:
```
<matplotlib.legend.Legend at 0x11737b08d90>
```



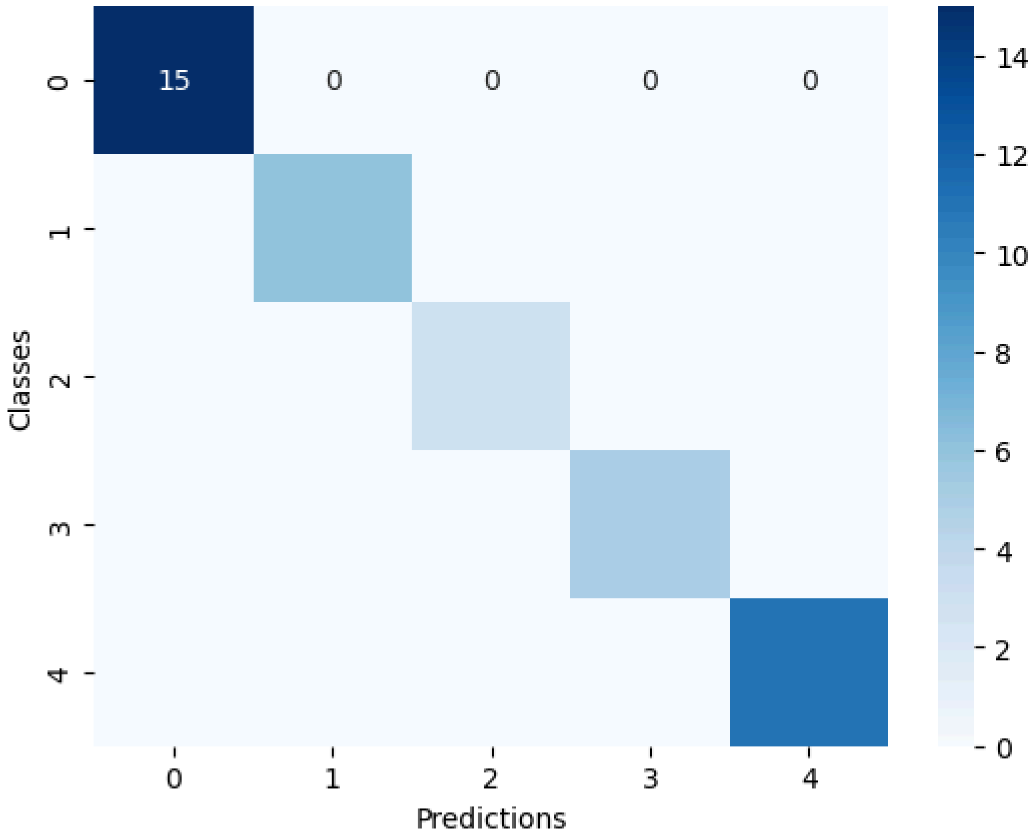
4. Test the model with the testset

In [74]:
```
cross_val_score(forest_clf, xtest, ytest, cv=3, scoring="accuracy")
```

Out[74]:
```
array([0.78571429, 0.92307692, 0.84615385])
```

In [75]:
```
ytest_pred = forest_clf.predict(xtest)
conf_matrix = confusion_matrix(ytest, ytest_pred)
sns.heatmap(data=conf_matrix, annot=True, cmap="Blues")
plt.xlabel("Predictions")
plt.ylabel("Classes")
```

Out[75]:
```
Text(50.722222222222214, 0.5, 'Classes')
```


Classes
Predictions