Exploration of Neural Networks and Machine Learning in Drug Classification

Author: Arshiya Khurmi

This project explores the application of machine learning techniques to the problem of drug classification based on patient-specific clinical features. Using a structured dataset containing medical attributes such as age, sex, blood pressure, cholesterol level, and sodium-to-potassium ratio, the goal was to develop a predictive model that can recommend one of five possible drug treatments.

A range of supervised learning models were implemented and evaluated, including Decision Tree, Random Forest, Neural Network (multi-class logistic regression), and CatBoost. The models were assessed using performance metrics such as accuracy and F1 score, as well as confusion matrices to evaluate their per-class precision. The final model was integrated into an interactive clinical support application using Gradio, allowing users to input patient data and receive real-time drug recommendations with associated confidence scores.

Data Preprocessing & Exploration

Import Libraries and Load Data

In [1]:
```
import pandas as pd

# Load the dataset
df = pd.read_csv('drug200.csv')
```

Note: The dataset file is included in the submission. Ensure it is located in the same directory as the notebook when running the code!

In [46]:
```
import numpy as np
import matplotlib.pyplot as plt

# catboost
from catboost import CatBoostClassifier

# sklearn
import sklearn.metrics
from sklearn.metrics import confusion_matrix, accuracy_score, classificatior
```

```
from sklearn.preprocessing import StandardScaler, LabelEncoder, OneHotEncode
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.ensemble import RandomForestClassifier

# Tensorflow.keras.x
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Input
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical

# Gradio
import gradio as gr
# Ignore warnings
import warnings
warnings.filterwarnings('ignore')
#Seaborn
import seaborn as sns
```

Data Exploration

To better understand the dataset prior to model development, I conducted a preliminary data exploration. This included inspecting data types, distributions, and potential imbalances. These insights informed preprocessing steps such as encoding categorical variables and scaling numerical features, helping to minimize training issues and improve model performance. Additionally, I visualized key features to identify patterns and support future interpretability.

In [52]:
```
df.head(5)
```

Out[52]:

	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 [58]:
```
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
```

Checking Missing Values

In [61]:
```
df.isnull().sum()
```

Out[61]:
```
Age            0
Sex            0
BP             0
Cholesterol    0
Na_to_K        0
Drug           0
dtype: int64
```

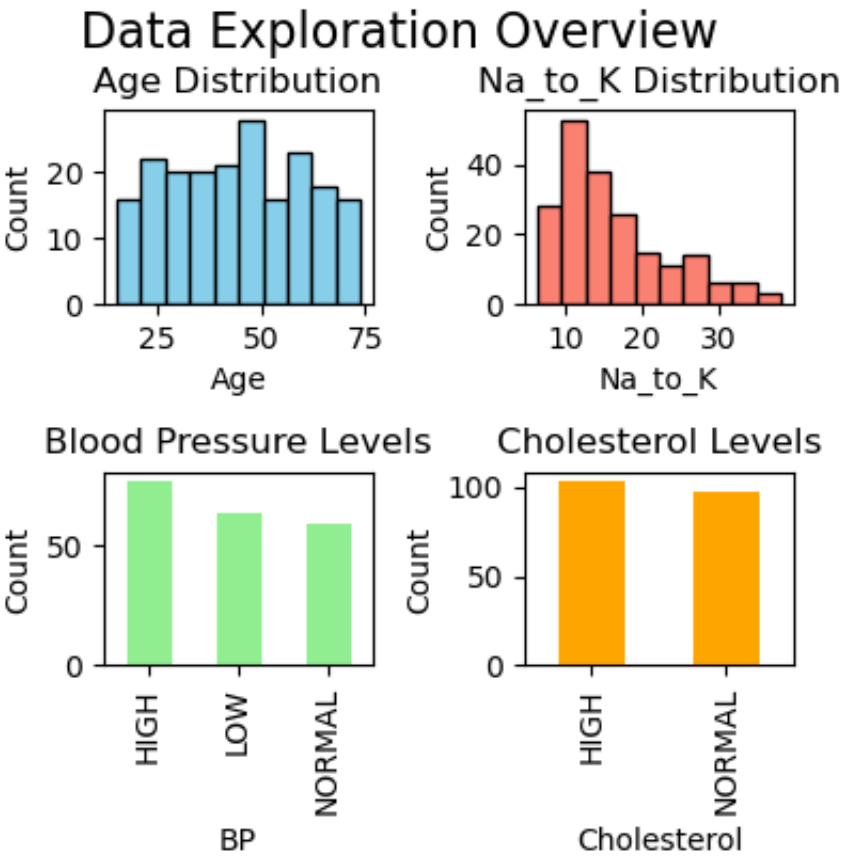
Encode Categorical Variables

In [96]:
```
df_encoded = pd.get_dummies(df, columns=['Sex', 'BP', 'Cholesterol'], drop_f
# Features and target
X = df_encoded.drop('Drug', axis=1)
y = df_encoded['Drug']

# Encode target labels
le = LabelEncoder()
y_encoded = le.fit_transform(y)
y_onehot = to_categorical(y_encoded)

X_train, X_test, y_train, y_test = train_test_split(X, y_onehot, test_size=0
```

Data Visualization

In [90]:
```
fig, axs = plt.subplots(2, 2, figsize=(4, 4))

# Histogram for Age
axs[0, 0].hist(df['Age'], bins=10, color='skyblue', edgecolor='black')
axs[0, 0].set_title('Age Distribution')
axs[0, 0].set_xlabel('Age')
```

```
axs[0, 0].set_ylabel('Count')

# Histogram for Na_to_K
axs[0, 1].hist(df['Na_to_K'], bins=10, color='salmon', edgecolor='black')
axs[0, 1].set_title('Na_to_K Distribution')
axs[0, 1].set_xlabel('Na_to_K')
axs[0, 1].set_ylabel('Count')

# Bar chart for BP
df['BP'].value_counts().plot(kind='bar', ax=axs[1, 0], color='lightgreen')
axs[1, 0].set_title('Blood Pressure Levels')
axs[1, 0].set_xlabel('BP')
axs[1, 0].set_ylabel('Count')

# Bar chart for Cholesterol
df['Cholesterol'].value_counts().plot(kind='bar', ax=axs[1, 1], color='orang
axs[1, 1].set_title('Cholesterol Levels')
axs[1, 1].set_xlabel('Cholesterol')
axs[1, 1].set_ylabel('Count')

plt.tight_layout()
plt.suptitle("Data Exploration Overview", fontsize=16, y=1.03)
plt.show()
```



To complete the analysis and gain a clearer understanding of the data distribution, I

visualized key features using various plots generated with Matplotlib. These visualizations provided insight into the spread and characteristics of the dataset, helping to guide model development.

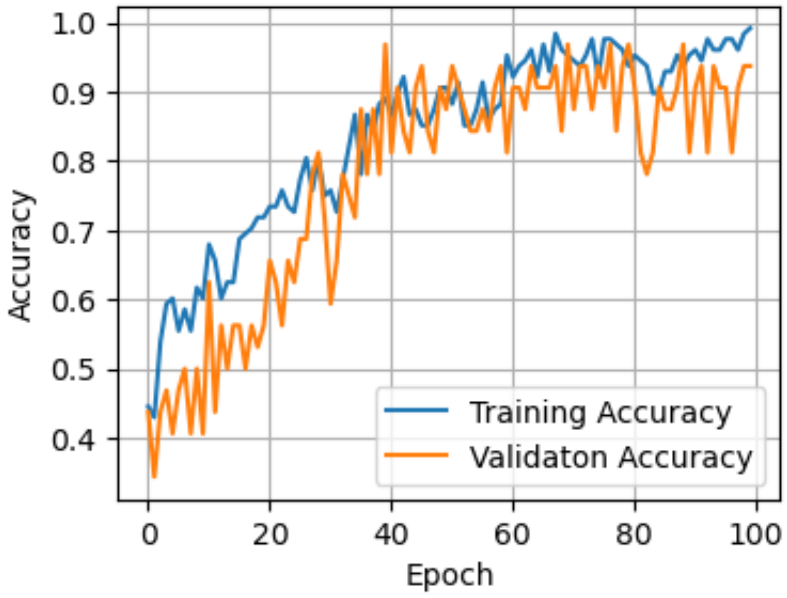
Model Training

Logistic Regression

In [7]:
```
# Multi-Class Logistic Regression - Neural Network
model = Sequential()
model.add(Dense(32, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(5, activation='softmax', input_shape=(X_train.shape[1],)))

# Compile Model
model.compile(optimizer=Adam(learning_rate=0.01), loss='categorical_crossent
```

In [9]:
```
# Measure Accuracy
history = model.fit(X_train, y_train, epochs=100, batch_size=16, validation_
```

In [115…
```
# Plot
plt.figure(figsize=(4, 3))
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validaton Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)
plt.show()
```



In [13]:
```
# Print training accuracy
final_train_acc = history.history['accuracy'][-1]
final_val_acc = history.history['val_accuracy'][-1]

print(f"Final Training Accuracy: {final_train_acc * 100:.2f}%")
print(f"Final Validation Accuracy: {final_val_acc * 100:.2f}%")
```

```
Final Training Accuracy: 99.22%
Final Validation Accuracy: 93.75%
```

To perform multi-class classification for drug recommendation, I implemented a logistic regression model using a fully connected neural network with a softmax-activated output layer. The model architecture consists of two hidden layers with ReLU activations (32 and 16 neurons, respectively), followed by a softmax output layer with 5 neurons corresponding to the five drug classes. The model was compiled using the Adam optimizer and trained with the categorical_crossentropy loss function, which is appropriate for multi-class classification tasks.

The figure above visualizes the training and validation accuracy across epochs. The training curve shows a steady increase, eventually plateauing at 99.22%. The validation accuracy, although slightly more volatile, follows a similar trend and converges to 93.75%, indicating strong generalization performance and no major signs of overfitting.

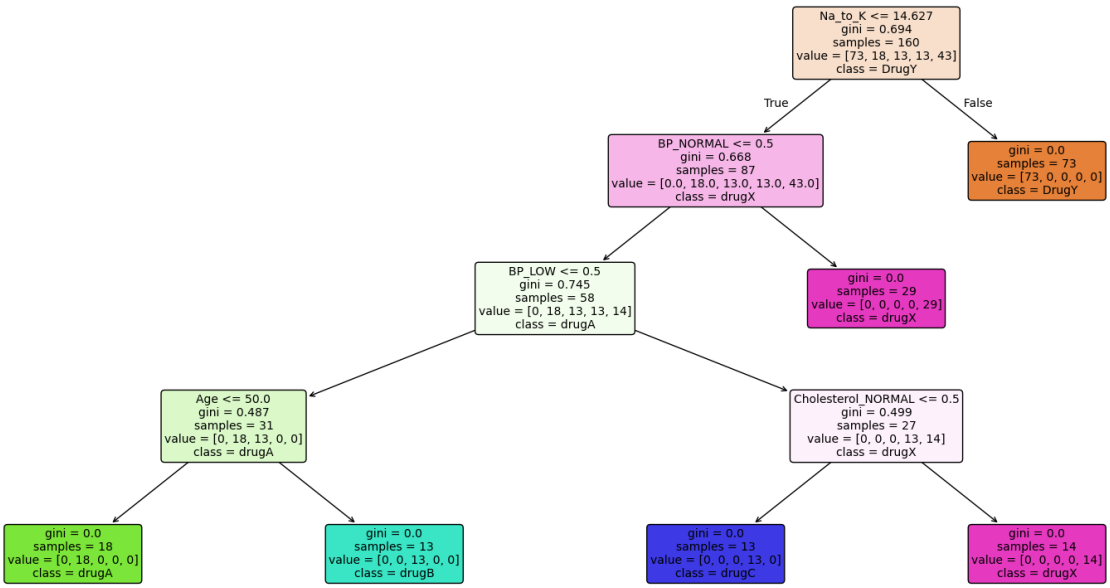
These results demonstrate that the neural network successfully learned to separate the five classes based on patient features. The small parity between training and validation performance suggests that the model is well-tuned and not overfitting, while the softmax output layer provides probability distributions. This is particularly well-suited for drug recommendation scenarios, where a clinician must evaluate multiple treatment options and determine the most appropriate choice based on patient-specific inputs such as age, blood pressure, cholesterol levels, and sodium-to-potassium ratio.

Decision Tree

In [16]:
```
# Convert one-hot back to single-label format
y_train_labels = np.argmax(y_train, axis=1)
y_test_labels = np.argmax(y_test, axis=1)
```

In [18]:
```
# Applying decision tree classifier
dt = DecisionTreeClassifier(max_depth=4, random_state=42)
dt.fit(X_train, y_train_labels)

# Predict
y_pred_dt = dt.predict(X_test)
```

In [50]:
```
# Plotting decision tree
plt.figure(figsize=(20, 10))
plot_tree(dt, feature_names=X.columns, class_names=le.classes_, filled=True,
plt.show()
```



A decision tree classifier was implemented to serve as an interpretable model. The tree was constrained to a maximum depth of 4 to avoid overfitting and maintain explainability. The tree visualization clearly shows how the model prioritizes the Na_to_K ratio as the root node, with subsequent splits on BP, Age, and Cholesterol, aligning well with known medical correlations for treatment recommendations. For instance, low sodium-to-potassium ratios and specific blood pressure values are frequently used to differentiate between drug types.

Random Forest

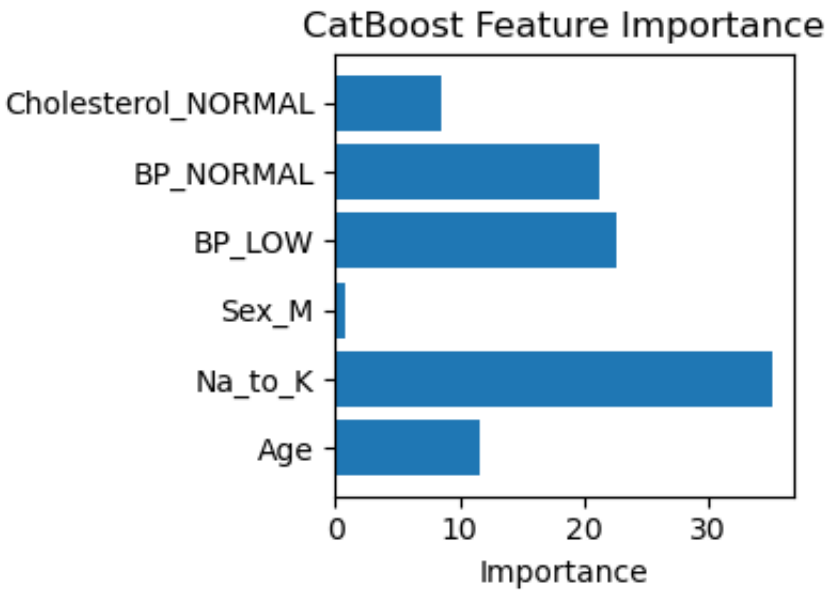
In [22]:
```
# Train random forest
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train_labels)

# Predict
y_pred_rf = rf.predict(X_test)
```

To improve classification performance and reduce overfitting risk, I implemented a Random Forest classifier consisting of 100 decision trees with a fixed random seed for reproducibility. Random forests aggregate the predictions of multiple decision trees, enhancing generalization by reducing the variance commonly seen in single-tree models.

CATBOOST

In [26]:
```
# Initialize CatBoost
cat_model = CatBoostClassifier(verbose=0, random_state=42)
cat_model.fit(X_train, y_train_labels)

# Predict
y_pred_cat = cat_model.predict(X_test)
```

In [117…
```
# Feature Importance
feature_importances = cat_model.get_feature_importance()
plt.figure(figsize=(4, 3))
plt.barh(X.columns, feature_importances)
plt.xlabel("Importance")
plt.title("CatBoost Feature Importance")
plt.tight_layout()
plt.show()
```



As a final model, I implemented a CatBoost classifier which is a high-performance gradient boosting algorithm well-suited for medical data. The feature importance plot revealed that Na_to_K was the most significant predictor. These findings align with both clinical expectations and prior literature as sodium-to-potassium ratio is a strong indicator in pharmacological decisions. Although CatBoost is not as interpretable as a decision tree, it can provide reliable insights. This combination of performance, interpretability, and flexibility makes CatBoost a great choice for real-world use in clinical decision support systems.

Accuracy Testing

Finding the Best Method

In [65]:
```
# Store results here
results = []

# Evaluation function
def evaluate_model(name, model, X_test, y_test, le, from_nn=False):
    # Neural Net model
    if from_nn:
        y_pred = np.argmax(model.predict(X_test), axis=1)
        y_true = np.argmax(y_test, axis=1)
    else:
        y_pred = model.predict(X_test)
        y_true = np.argmax(y_test, axis=1) if len(y_test.shape) > 1 else y_t

    acc = accuracy_score(y_true, y_pred)
    f1 = f1_score(y_true, y_pred, average='weighted')

    results.append({"Model": name, "Accuracy": acc, "F1 Score": f1,
                    "Classification Report": classification_report(y_true, y
    })

# Evaluate all models
evaluate_model("Decision Tree", dt, X_test, y_test, le)
evaluate_model("Random Forest", rf, X_test, y_test, le)
evaluate_model("CatBoost", cat_model, X_test, y_test, le)
evaluate_model("Neural Network", model, X_test, y_test, le, from_nn=True)

# Format summary table
summary_table = pd.DataFrame([{
    "Model": r["Model"],
    "Accuracy (%)": round(r["Accuracy"] * 100, 2),
    "F1 Score (%)": round(r["F1 Score"] * 100, 2)
} for r in results])

summary_table = summary_table.sort_values(by="Accuracy (%)", ascending=False

# Display summary
print("\n Model Comparison Summary:")
print(summary_table)
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step

 Model Comparison Summary:
           Model  Accuracy (%)  F1 Score (%)
0   Decision Tree          97.5         97.47
1   Random Forest          97.5         97.47
2        CatBoost          97.5         97.47
3  Neural Network          95.0         95.06
```

To identify the most suitable classifier for drug prediction, all 4 models were evaluated and compared using accuracy and F1 score. These metrics were chosen to assess overall correctness and class-wise balance in multi-class prediction.

The decision tree model achieved a test accuracy of 97.50%, which is unexpectedly high given the model's simplicity and depth limitation. While this suggests strong class separability within the dataset, it may also be a reflection of the decision tree's ability to leverage highly discriminative features (like Na_to_K) at early splits. Despite its high accuracy, decision trees are prone to overfitting in larger, noisier datasets, and lack the robustness of ensemble methods. However, in this case, the decision path visualization offers valuable clinical insight and enhances trust in the model's decision-making process

The Random Forest model achieved an accuracy of 97.50% on the test set, matching the decision tree's performance but offering better robustness. This result suggests that ensemble averaging played a key role in stabilizing predictions and reducing the effect of noisy splits that might have otherwise led to misclassifications.

The CatBoost model achieved a test accuracy of 97.50%, matching the performance of the decision tree and random forest classifiers. However, CatBoost offers greater robustness and generalization by sequentially building decision trees in a boosting framework, which helps correct previous misclassifications (Mridha et al., 2024).

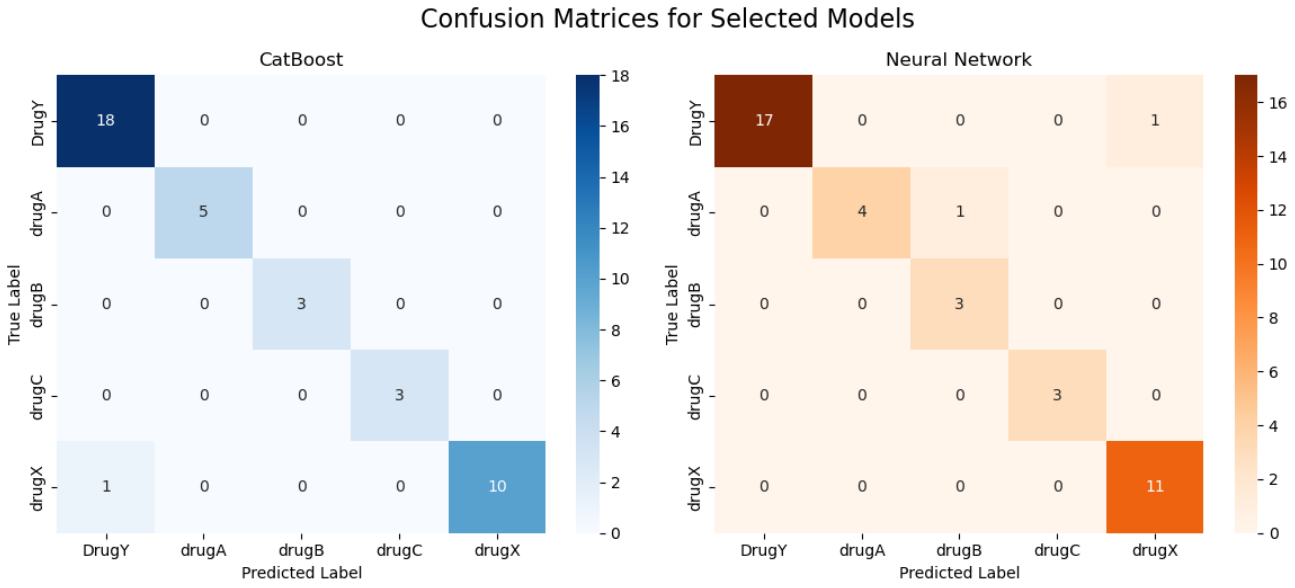
While all models performed well, CatBoost was selected as the final model for use in the clinical support tool below due to its model explainability in clinical contexts. This decision is further supported by work by Mridha et al. (2024).

Comparative Confusion Matrix

In [48]:
```
# For CatBoost
y_pred_cat = cat_model.predict(X_test)
y_true_labels = np.argmax(y_test, axis=1)

# For Neural Network
y_pred_nn_probs = model.predict(X_test)
```

```
y_pred_nn = np.argmax(y_pred_nn_probs, axis=1)

cm_cat = confusion_matrix(y_true_labels, y_pred_cat)
cm_nn = confusion_matrix(y_true_labels, y_pred_nn)

fig, axs = plt.subplots(1, 2, figsize=(12, 5))

# CatBoost confusion matrix
sns.heatmap(cm_cat, annot=True, fmt='d', cmap='Blues', ax=axs[0],
            xticklabels=le.classes_, yticklabels=le.classes_)
axs[0].set_title("CatBoost")

# Neural Network confusion matrix
sns.heatmap(cm_nn, annot=True, fmt='d', cmap='Oranges', ax=axs[1],
            xticklabels=le.classes_, yticklabels=le.classes_)
axs[1].set_title("Neural Network")

# Layout
for ax in axs:
    ax.set_xlabel("Predicted Label")
    ax.set_ylabel("True Label")

plt.tight_layout()
plt.suptitle("Confusion Matrices for Selected Models", fontsize=16, y=1.05)
plt.show()
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
```



The confusion matrix comparison reinforces CatBoost's superior class-level accuracy, particularly its ability to perfectly classify less-represented classes. In contrast, the neural network demonstrated slightly more variability, particularly on DrugA and DrugY where small errors appear. This validates the decision to use CatBoost.

Mini Clinical Drug Classifier Tool

In [94]:
```python
def predict_drug(age, sex, bp, chol, na_to_k):
    input_data = {
        'Age': age,
        'Na_to_K': na_to_k,
        'Sex_Male': 1 if sex.lower() == 'male' else 0,
        'BP_High': 1 if bp.lower() == 'high' else 0,
        'BP_Normal': 1 if bp.lower() == 'normal' else 0,
        'Cholesterol_High': 1 if chol.lower() == 'high' else 0
    }
    input_df = pd.DataFrame([input_data], columns=cat_model.feature_names_)

    # Predict class and probabilities
    pred_class = cat_model.predict(input_df)
    pred_label = le.inverse_transform([int(pred_class[0])])[0]
    probs = cat_model.predict_proba(input_df)[0]

    # Format output
    prob_output = "\n".join([
        f"{le.classes_[i]}: {p*100:.2f}%" for i, p in enumerate(probs)
    ])
    return f" Predicted Drug: {pred_label}", prob_output
```

In [ ]:
```python
# Creating interface – Run with jupyter notebook
gr.Interface(
    fn=predict_drug,
    inputs=[
        gr.Slider(15, 80, label="Age"),
        gr.Radio(["Male", "Female"], label="Sex"),
        gr.Radio(["Low", "Normal", "High"], label="Blood Pressure"),
        gr.Radio(["Normal", "High"], label="Cholesterol"),
        gr.Slider(5.0, 40.0, label="Na_to_K Ratio")
    ],
    outputs=[
        gr.Textbox(label="Prediction"),
        gr.Textbox(label="Class Probabilities")
    ],
    title="Mini Clinical Drug Recommender using CatBoost",
    description="Select patient characteristics to predict a recommended dru
).launch(share=True)
```

As the culminating component of this project, I deployed a functional Mini Clinical Drug Recommender Tool using Gradio. Powered by the CatBoost classifier, the app allows users to input five key patient features to predict a recommended drug from five possible categories. In addition to the prediction, the tool displayed the associated class probabilities, offering transparency in model confidence. This tool is a real-world

application of the machine learning pipeline developed throughout the project.

Running the app: To launch the interactive drug classification tool:

1. Run the code cell containing the Gradio interface. (Note: This cell is not displayed in the report, but will produce an interface when executed.).
2. Click either the local or public link that appears to open the app in a new browser tab.

If running in a restricted environment where share=True does not work, simply remove the share=True argument and use: demo.launch().

Conclusion

This project demonstrates the full machine learning lifecycle, from data preprocessing to model training, comparative analysis, and real-world deployment. The final tool mirrors a clinical decision support system, highlighting how machine learning can assist in healthcare settings by automating decision-making based on structured patient data.

References

Mridha, K., Bappon, S. D., Sabuj, S. M., Sarker, T., & Ghosh, A. (2024). Explainable machine learning for drug classification. Lecture Notes in Electrical Engineering, 673–683. https://doi.org/10.1007/978-981-99-8661-3_48

Tripathi, P. (2021). *Drug Classification Dataset*. Kaggle. https://www.kaggle.com/datasets/prathamtripathi/drug-classification