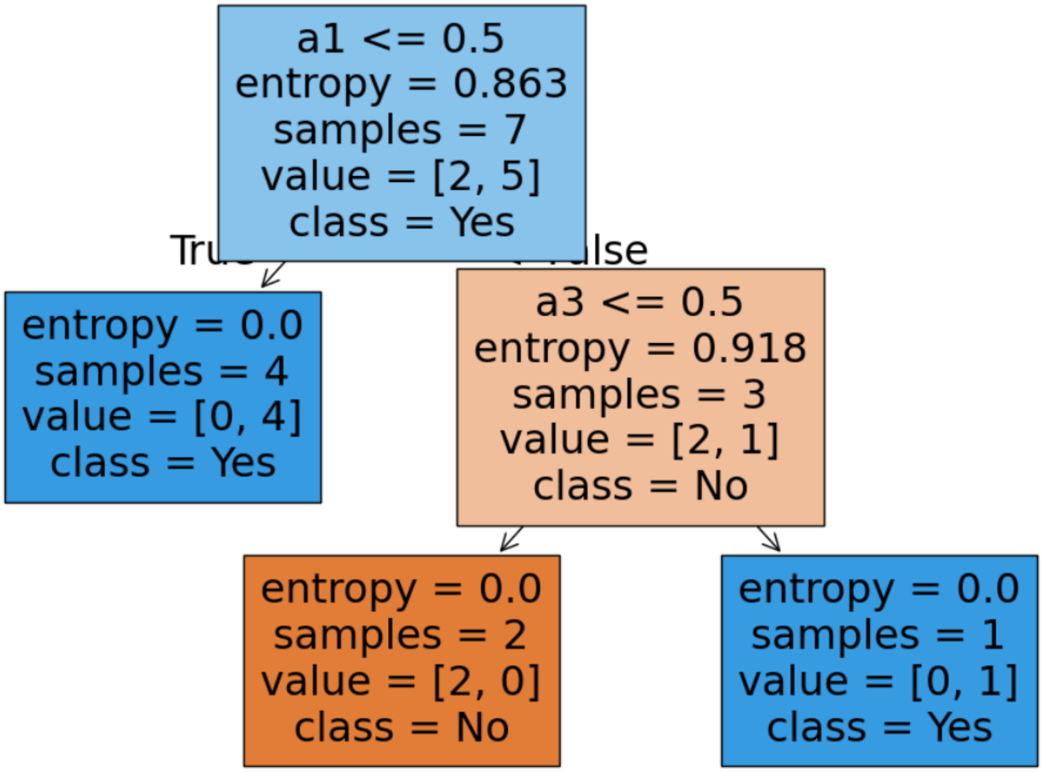
Lab 4

DECISION TREE:

In [1]:
```python
#decision_tree
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report

# Create the dataset
data = {
    'a1': [True, True, False, False, False, True, True, True, False, False],
    'a2': ['Hot', 'Hot', 'Hot', 'Cool', 'Cool', 'Cool', 'Hot', 'Hot', 'Cool', 'Cool'],
    'a3': ['High', 'High', 'High', 'Normal', 'Normal', 'High', 'High', 'Normal', 'Normal', 'High'],
    'Classification': ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'No', 'Yes', 'Yes', 'Yes']
}

data

# Convert to DataFrame
df = pd.DataFrame(data)

# Convert categorical data to numerical data
label_encoders = {}
for column in df.columns:
    le = LabelEncoder()
    df[column] = le.fit_transform(df[column])
    label_encoders[column] = le

# Split the dataset into features and target
X = df.drop('Classification', axis=1)
y = df['Classification']
```

```python
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Initialize the Decision Tree Classifier with entropy as the criterion
clf = DecisionTreeClassifier(criterion='entropy')

# Train the classifier
clf.fit(X_train, y_train)

# Make predictions
y_pred = clf.predict(X_test)

# Evaluate the classifier
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
print(classification_report(y_test, y_pred, target_names=['No', 'Yes']))

# Optionally, visualize the decision tree
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

plt.figure(figsize=(12,8))
plot_tree(clf, filled=True, feature_names=X.columns, class_names=['No', 'Yes'])
plt.show()
```

```
Accuracy: 1.00
              precision    recall  f1-score   support

          No       1.00      1.00      1.00         2
         Yes       1.00      1.00      1.00         1

    accuracy                           1.00         3
   macro avg       1.00      1.00      1.00         3
weighted avg       1.00      1.00      1.00         3
```



In [5]:

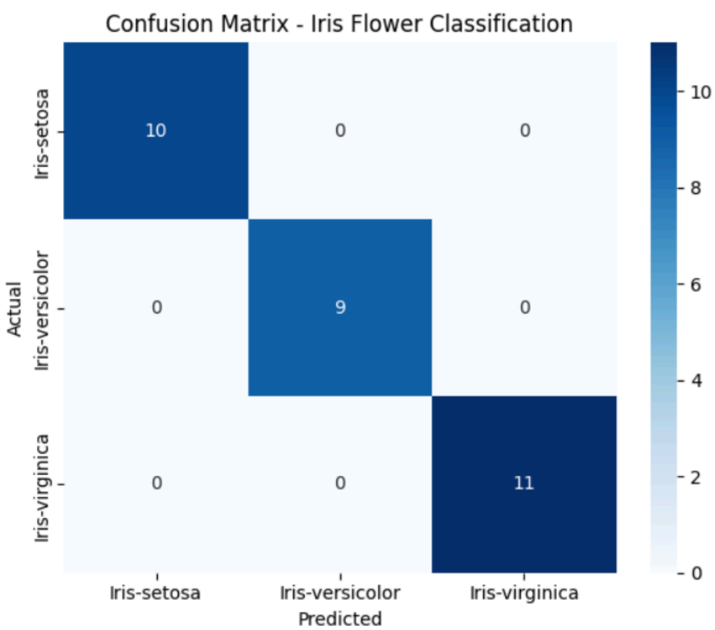
```
#iris_dataset
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
from google.colab import files

# Upload the dataset
print("Please upload 'iris.csv'")
uploaded = files.upload()

# Load the dataset
df = pd.read_csv(next(iter(uploaded)))
print("\nDataset Preview:")
print(df.head())

# Define features (X) and target (y)
X = df.drop('species', axis=1)  # Features
y = df['species']  # Target variable

# Split into train (80%) and test (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Build and train the Decision Tree classifier
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)

# Predict on test data
y_pred = model.predict(X_test)
```

```
model.fit(X_train, y_train)

# Predict on test data
y_pred = model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"\nAccuracy Score: {accuracy:.2f}")

# Display confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot the confusion matrix
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=model.classes_, yticklabels=model.classes_)
plt.title('Confusion Matrix - Iris Flower Classification')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.tight_layout()
plt.show()
```

```
Please upload 'iris.csv'
```

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving iris (1).csv to iris (1).csv

Dataset Preview:
   sepal_length  sepal_width  petal_length  petal_width      species
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa

Accuracy Score: 1.00

 Accuracy Score: 1.00
```



In [4]:

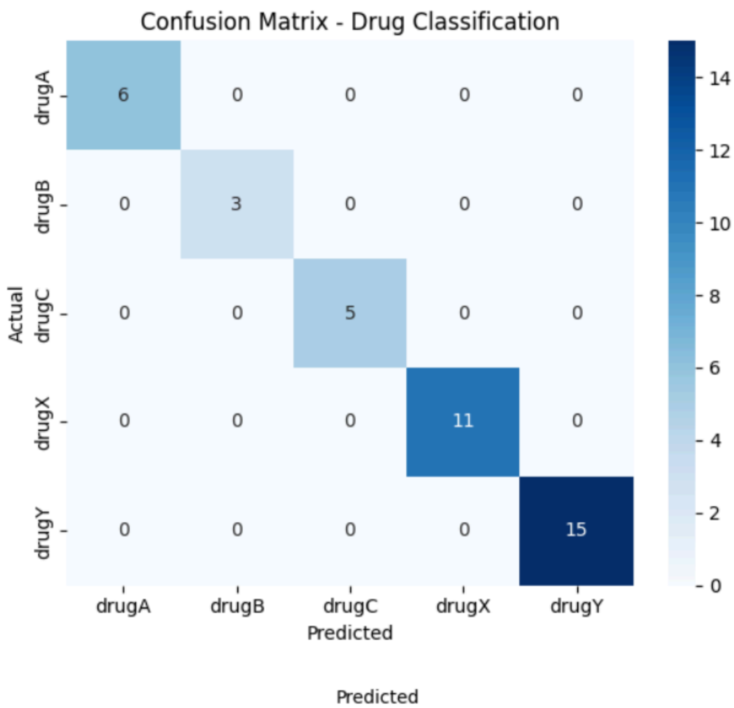
```python
#drug_dataset
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
from google.colab import files

# Upload the dataset
print("Please upload 'drug.csv'")
uploaded = files.upload()

# Load the dataset
df = pd.read_csv(next(iter(uploaded)))
print("\nDataset Preview:")
print(df.head())

# Preprocess data if necessary (e.g., convert categorical variables to numerical)
# Convert categorical 'Sex', 'BP', and 'Cholesterol' columns to numerical using encoding
df['Sex'] = df['Sex'].map({'M': 0, 'F': 1})  # Male = 0, Female = 1
df['BP'] = df['BP'].map({'LOW': 0, 'NORMAL': 1, 'HIGH': 2})  # Encoding BP levels
df['Cholesterol'] = df['Cholesterol'].map({'NORMAL': 0, 'HIGH': 1})  # Encoding Cholesterol levels

# Define features (X) and target (y)
X = df.drop('Drug', axis=1)
y = df['Drug']

# Split into train (80%) and test (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Build and train the Decision Tree classifier
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)
```

```python
# Predict on test data
y_pred = model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"\nAccuracy Score: {accuracy:.2f}")

# Display confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot the confusion matrix
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=model.classes_, yticklabels=model.classes_)
plt.title('Confusion Matrix - Drug Classification')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.tight_layout()
plt.show()
```

```
Please upload 'drug.csv'
```

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving drug.csv to drug.csv

Dataset Preview:
   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

Accuracy Score: 1.00
```



In [3]:

```python
#petrol_consumption
# Import required libraries
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeRegressor, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error
import matplotlib.pyplot as plt
from google.colab import files

# Upload the dataset
print("Please upload 'petrol_consumption.csv'")
uploaded = files.upload()

# Load the dataset
df = pd.read_csv(next(iter(uploaded)))
print("\nDataset Preview:")
print(df.head())

# Define features (X) and target (y)
X = df.drop('Petrol_Consumption', axis=1)
y = df['Petrol_Consumption']

# Split into train (80%) and test (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Build and train the regression tree model
model = DecisionTreeRegressor(random_state=42)
model.fit(X_train, y_train)

# Predict on test data
y_pred = model.predict(X_test)
```

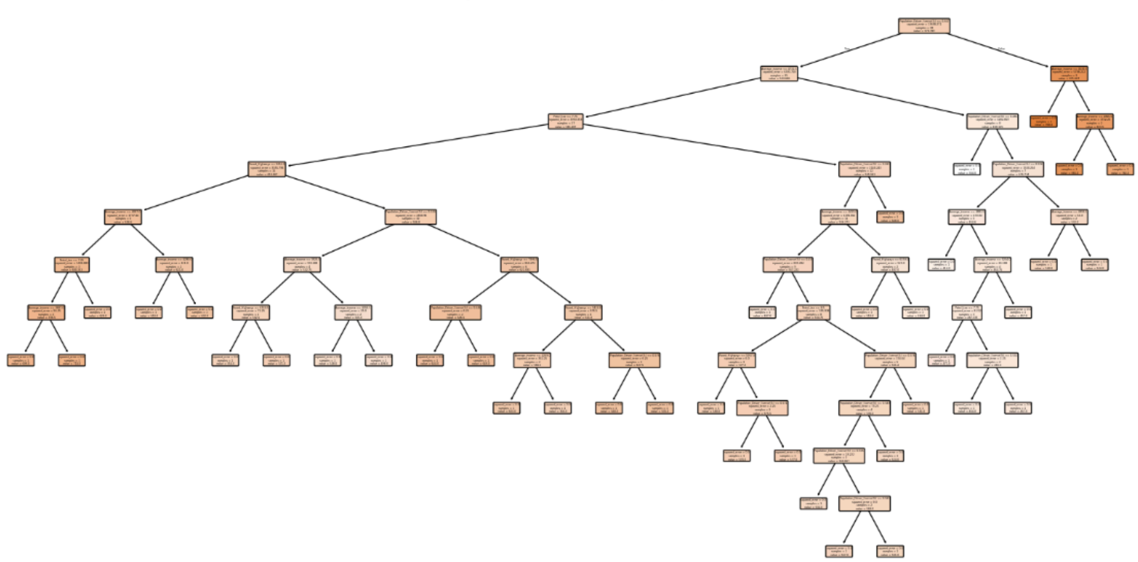
```python
# Build and train the regression tree model
model = DecisionTreeRegressor(random_state=42)
model.fit(X_train, y_train)

# Predict on test data
y_pred = model.predict(X_test)

# Calculate error metrics
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)

# Display the error metrics
print(f"\n📊 Evaluation Metrics:")
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")

# Visualize the regression tree
plt.figure(figsize=(20,10))
plot_tree(model, feature_names=X.columns, filled=True, rounded=True)
plt.title("Regression Tree for Petrol Consumption")
plt.show()
```

```
Please upload 'petrol_consumption.csv'
```

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving petrol_consumption.csv to petrol_consumption (1).csv

Dataset Preview:
   Petrol_tax  Average_income  Paved_Highways  Population_Driver_licence(%)  \
0         9.0            3571            1976                         0.525
1         9.0            4092            1250                         0.572
2         9.0            3865            1586                         0.580
3         7.5            4870            2351                         0.529
4         8.0            4399             431                         0.544


   Petrol_Consumption
0                 541
1                 524
2                 561
3                 414
4                 410

📊 Evaluation Metrics:
Mean Absolute Error (MAE): 94.30
Mean Squared Error (MSE): 17347.70
Root Mean Squared Error (RMSE): 131.71
```

Regression Tree for Petrol Consumption



27/3/25
Monday

Lab-4

Consider the following dataset calculate Entropy and Information Gain w.r.t to target variable "classification". Identify whether the splitting node should be a_2 or a_3 attribute.

instance	a_2	a_3	classification
1	Hot	High	NO
2	Hot	High	NO
6	Cool	High	NO
7	Hot	High	NO
8	Hot	Normal	Yes

Gain $(S, a_1) = 0.6099$ - maximum Gain
Gain $(S, a_2) = 0.1245$
Gain $(S, a_3) = 0.4199$

a_1
True / \ False
1, 2, 6, 7, 8 — 3, 4, 5, 9, 10

Attribute a_2
value (a_2) = Hot, cool
$S_{a_1} = [1+, 4-]$
Entropy $(S.a_1) = -\frac{1}{5}\log_2\frac{1}{5} - \frac{4}{5}\log_2\frac{4}{5}$

$= 0.7219$

$S_{Hot} = [1+; 3-]$ Entropy $= \frac{1}{4}\log_2\frac{1}{4} - \frac{3}{4}\log_2\frac{3}{4}$

$= 0.6112$

$S_{cool} \leftarrow [0+, 1-]$ Entropy$(S_{cool}) = 0.0$

Gain ← [0+, 1−]

$Gain(S, a_2) = Entropy(S) - \sum_{v \in \{Hot, Cool\}} \frac{|S_v|}{|S|} Entropy(S_v)$

$Gain(S, a_2) = Entropy(S) - 4/5\ Entropy(S_{Hot}) - 1/5\ Entropy(S_{Cool})$

$Gain(S, a_2) = 0.9709 - 4/5 \times 0.8112 - 1/5 \times 0.0 = 0.3219$

Attribute : a_3

Value (a_3) = High ; Normal

$S_{a_1} = [1+, 4-]$ Entropy $[S_{a_1}] = -1/5 \log_2 1/5 - 4/5 \log_2 4/5$
$= 0.7219$

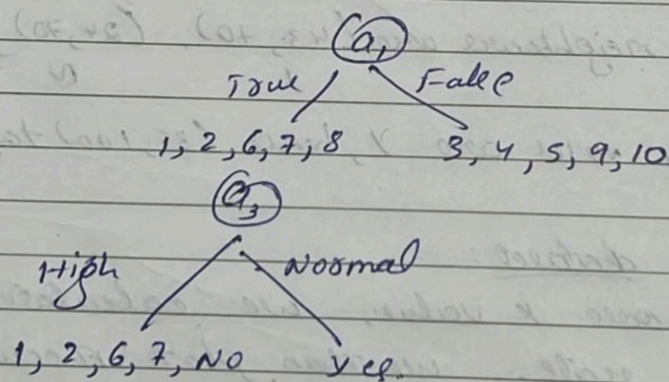
$S_{High} = [0+, 4-]$ entropy $(S_{High}) = 0.0$

$S_{normal} = [1+, 0-]$ Entropy $(S_{normal}) = 0.0$

$Gain(S, a_3) = Entropy(S) - \sum_{v \in (High, Normal)} \frac{|S_v|}{|S|} Entropy(S_v)$

$Gain(S, a_3) = Entropy(S) - 4/5\ Entropy(S_{High}) - \frac{1}{5}\ Entropy$

$Gain(S, a_3) = 0.9709 - 4/5 \times 0.0 - 1/5 \times 0.0 = 0.7219$



(3) For "iris.csv" data.