Lab 5: Introducing Classification

Objectives:

- To gain hands-on experience classifying small dataset
- To implement concepts related to Decision Tree classifier (i.e. Entropy, Information Gain), along with using existing libraries.

```
# Run this cell if you use Colab
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Code it yourself

```
import pandas as pd
import numpy as np

# Read the data
df = pd.read_csv('./toy_data.csv')
df
```

	age	income	student	credit rating	buys computer
0	<=30	high	no	fair	no
1	<=30	high	no	excellent	no
2	31-40	high	no	fair	yes
3	>40	medium	no	fair	yes
4	>40	low	yes	fair	yes
5	>40	low	yes	excellent	no
6	31-40	low	yes	excellent	yes
7	<=30	medium	no	fair	no
8	<=30	low	yes	fair	yes
9	>40	medium	yes	fair	yes
10	<=30	medium	yes	excellent	yes
11	31-40	medium	no	excellent	yes
12	31-40	high	yes	fair	yes
13	>40	medium	no	excellent	no

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   age            14 non-null     object
 1   income         14 non-null     object
 2   student        14 non-null     object
 3   credit rating  14 non-null     object
 4   buys computer  14 non-null     object
dtypes: object(5)
memory usage: 692.0+ bytes
None
```

TODO: Write functions to compute Gain and Entropy, as discussed in the lecture.

```
import math


# Write your code here
entropy_t = pd.Series(df['buys computer'].value_counts().values/df['buys computer'].count())
entropy_t = -entropy_t.apply(lambda x: x*math.log2(x)).sum()
entropy_t
```

```
0.9402859586706311
```

```
ni_n = df.groupby('age')['buys computer'].count()/df['buys computer'].count()
entropy_age = (
    df.groupby('age')['buys computer'].value_counts()/df.groupby('age')['buys computer'].count()
    ).apply(lambda x: -x*math.log2(x)).groupby('age').sum()*ni_n

entropy_age
```

age	0
31-40	0.000000
<=30	0.346768
>40	0.346768

dtype: float64

```
gain_age = entropy_t - entropy_age.sum()
gain_age
```

```
0.24674981977443933
```

```
to_cal_attr = df.columns[:4]
gain_attr = {}
for attr in to_cal_attr:
    ni_n = df.groupby(attr)['buys computer'].count()/df['buys computer'].count()
    entropy_attr = (
        df.groupby(attr)['buys computer'].value_counts()/df.groupby(attr)['buys computer'].count()
        ).apply(lambda x: -x*math.log2(x)).groupby(attr).sum()*ni_n
    gain_attr[attr] = entropy_t - entropy_attr.sum()

gain_attr
```

```
{'age': 0.24674981977443933,
 'income': 0.02922256565895487,
 'student': 0.15183550136234159,
 'credit rating': 0.04812703040826949}
```

Using Libraries

Now that you know how to compute these values by yourselfs, now let's use some libraries.

Steps:

- Split the Data → Divide dataset into training (80%) and testing (20%).
- Train the Model → Fit a Decision Tree using the training data.
- Test the Model → Use the trained model to predict on test data.
- Evaluate Performance → Compare predictions with actual values (e.g., Accuracy Score).

Prepare features and labels.

```
# Features
features = df.drop('buys computer', axis=1)
features

# Alternatively, you can use this:
```

```
# features = df.iloc[:, :-1]
```

	age	income	student	credit rating
0	<=30	high	no	fair
1	<=30	high	no	excellent
2	31-40	high	no	fair
3	>40	medium	no	fair
4	>40	low	yes	fair
5	>40	low	yes	excellent
6	31-40	low	yes	excellent
7	<=30	medium	no	fair
8	<=30	low	yes	fair
9	>40	medium	yes	fair
10	<=30	medium	yes	excellent
11	31-40	medium	no	excellent
12	31-40	high	yes	fair
13	>40	medium	no	excellent

Next steps: Generate code with features | View recommended plots | New interactive sheet

```
# Labels (or Target)
labels = df['buys computer']
labels

# # Alternatively, you can use this:
# labels = df.iloc[:, [-1]]
```

	buys computer
0	no
1	no
2	yes
3	yes
4	yes
5	no
6	yes
7	no
8	yes
9	yes
10	yes
11	yes
12	yes
13	no

dtype: object

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree

# 1. Load the dataset
X = features.values  # Features
y = labels.values  # Target labels
```

```
# 2. Split the dataset into training (80%) and testing (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 3. Create and train a Decision Tree model with entropy criterion
clf = DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=42)
clf.fit(X_train, y_train)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-25-047a60d5fd52> in <cell line: 0>()
     13 # 3. Create and train a Decision Tree model with entropy criterion
     14 clf = DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=42)
---> 15 clf.fit(X_train, y_train)

                                 5 frames
/usr/local/lib/python3.11/dist-packages/sklearn/utils/_array_api.py in _asarray_with_order(array, dtype, order, copy, xp, device)
    837             array = numpy.array(array, order=order, dtype=dtype)
    838         else:
--> 839             array = numpy.asarray(array, order=order, dtype=dtype)
    840
    841         # At this point array is a NumPy ndarray. We convert it to an array

ValueError: could not convert string to float: '31-40'
```

Next steps: Explain error

There's an error:

ValueError: could not convert string to float: '31-40'

```
from sklearn.preprocessing import LabelEncoder

# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Apply Label Encoding for all categorical columns
df['age'] = label_encoder.fit_transform(df['age'])
df['income'] = label_encoder.fit_transform(df['income'])
df['student'] = label_encoder.fit_transform(df['student'])
df['credit rating'] = label_encoder.fit_transform(df['credit rating'])
df['buys computer'] = label_encoder.fit_transform(df['buys computer'])

# Display the encoded DataFrame
print(df)
```

```
    age  income  student  credit rating  buys computer
0     1       0        0              1              0
1     1       0        0              0              0
2     0       0        0              1              1
3     2       2        0              1              1
4     2       1        1              1              1
5     2       1        1              0              0
6     0       1        1              0              1
7     1       2        0              1              0
8     1       1        1              1              1
9     2       2        1              1              1
10    1       2        1              0              1
11    0       2        0              0              1
12    0       0        1              1              1
13    2       2        0              0              0
```

Let's check out an updated dataframe.

```
df
```

	age	income	student	credit rating	buys computer
0	1	0	0	1	0
1	1	0	0	0	0
2	0	0	0	1	1
3	2	2	0	1	1
4	2	1	1	1	1
5	2	1	1	0	0
6	0	1	1	0	1
7	1	2	0	1	0
8	1	1	1	1	1
9	2	2	1	1	1
10	1	2	1	0	1
11	0	2	0	0	1
12	0	0	1	1	1
13	2	2	0	0	0

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
X = df.drop('buys computer', axis=1)  # Features
y = df['buys computer']  # Target
```

Let's continue where we left off!

```
# 2. Split the dataset into training (80%) and testing (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 3. Create and train a Decision Tree model with entropy criterion
clf = DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=42)
clf.fit(X_train, y_train)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=42)
```

```
print(X_train.shape)
print(X_test.shape)
```

```
(11, 4)
(3, 4)
```

Now we're going build the Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier

# Initialize the Decision Tree classifier
clf = DecisionTreeClassifier(criterion='entropy', random_state=42)  # Using 'entropy' as the criterion

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

# Predict on the test set
y_pred = clf.predict(X_test)
```

And evaluate our model.

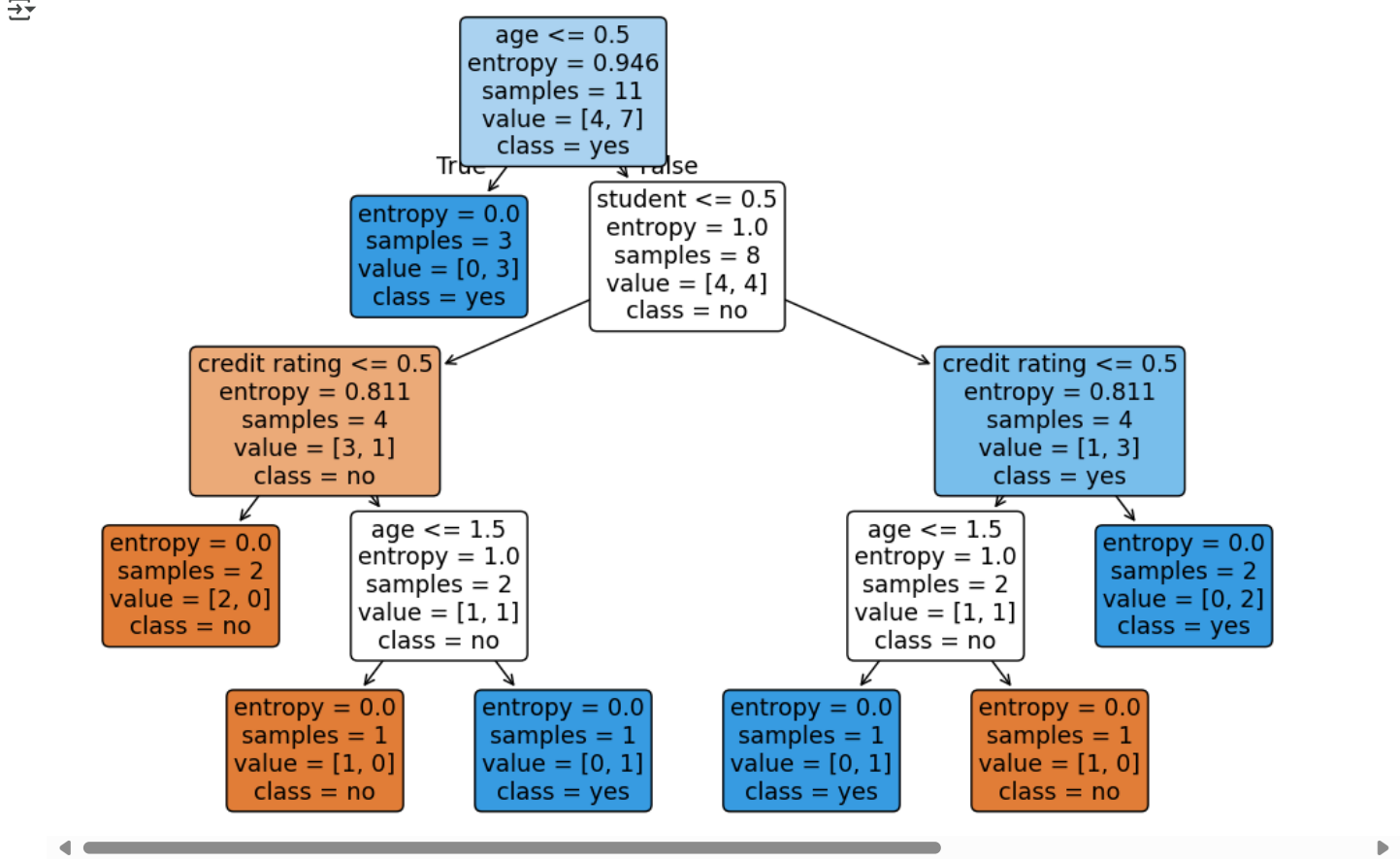
```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

```
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

# Classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))

# Confusion Matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

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

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         2

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

Confusion Matrix:
[[1 0]
 [0 2]]
```

And visualize our tree!

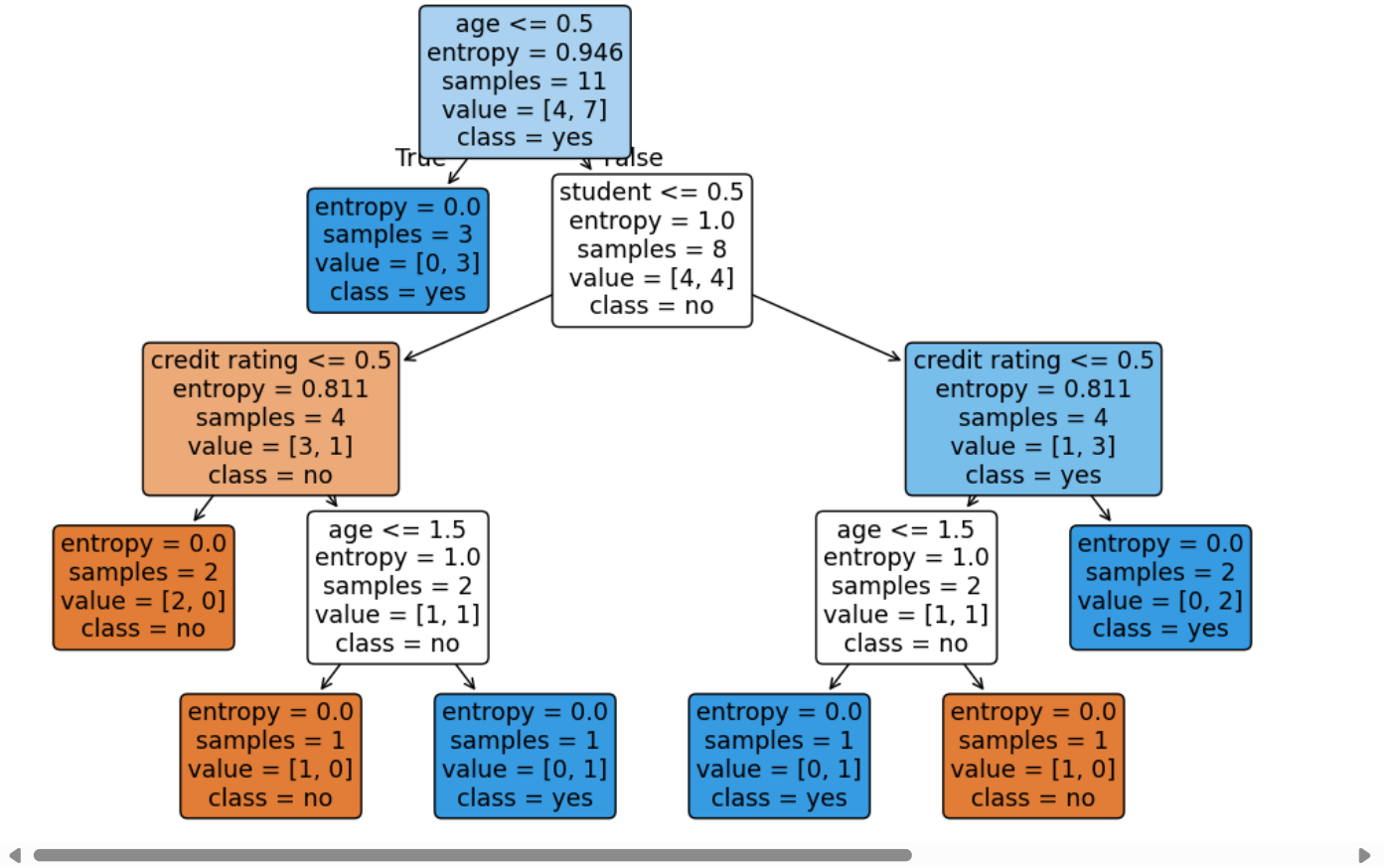
```
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

# Plot the decision tree
plt.figure(figsize=(12, 8))
plot_tree(clf, filled=True, feature_names=X.columns, class_names=['no', 'yes'], rounded=True)
plt.show()
```



Put them all together.

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

# data
data = pd.read_csv('./toy_data.csv')

df = pd.DataFrame(data)

# Encode categorical columns using LabelEncoder
label_encoder = LabelEncoder()
df['age'] = label_encoder.fit_transform(df['age'])
df['income'] = label_encoder.fit_transform(df['income'])
df['student'] = label_encoder.fit_transform(df['student'])
df['credit rating'] = label_encoder.fit_transform(df['credit rating'])
df['buys computer'] = label_encoder.fit_transform(df['buys computer'])

# Separate features (X) and target (y)
X = df.drop('buys computer', axis=1)
y = df['buys computer']

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

# Initialize the Decision Tree classifier
clf = DecisionTreeClassifier(criterion='entropy', random_state=42)

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

# Predict on the test set
y_pred = clf.predict(X_test)

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

# Classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))

# Confusion Matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

# Plot the decision tree
plt.figure(figsize=(12, 8))
plot_tree(clf, filled=True, feature_names=X.columns, class_names=['no', 'yes'], rounded=True)
plt.show()
```

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

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         2

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

Confusion Matrix:
[[1 0]
 [0 2]]
```



Is the output tree the same as what you calculated yourself? Explain in your own words why they are the same or different.

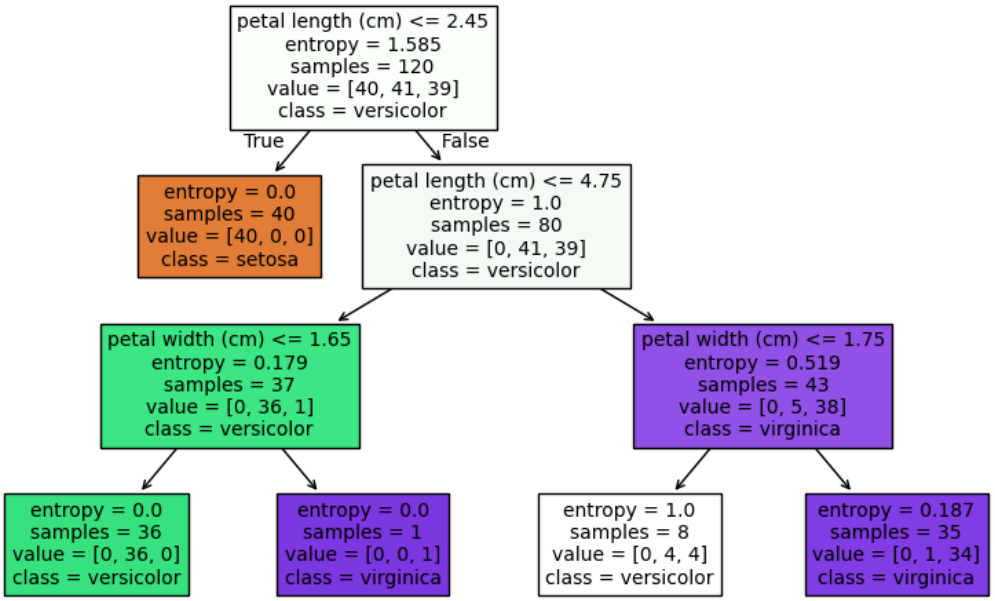
Ans: Nope, The output tree is different because the model was trained on only 80% of the data, but the manual calculation used all the data. This caused changes in entropy values, information gain, and the tree structure.

Another example, another dataset -- Iris

```
from sklearn.datasets import load_iris

# 1. Load the Iris dataset
iris = load_iris()
X = iris.data  # Features
y = iris.target  # Target labels

# 2. Split the dataset into training (80%) and testing (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 3. Create and train a Decision Tree model with entropy criterion
clf = DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=42)
clf.fit(X_train, y_train)

# 4. Make predictions on the test set
```

```
y_pred = clf.predict(X_test)

# 5. Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy:.2f}")

# 6. Visualize the Decision Tree
plt.figure(figsize=(10, 6))
plot_tree(clf, filled=True, feature_names=iris.feature_names, class_names=iris.target_names)
plt.show()
```

```
Model Accuracy: 1.00
```



Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.