Problem: We have online music library/app, when a new user registers, predicting what type of music they may like based on age and gender? Later we can recommend the same.

In [1]:
```
# Import the data
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

music_data = pd.read_csv("/Users/roja/Downloads/Python Mini Projects Using Jupyter/music.csv")
music_data
```

Out[1]:

	age	gender	genre
0	20	1	HipHop
1	23	1	HipHop
2	25	1	HipHop
3	26	1	Jazz
4	29	1	Jazz
5	30	1	Jazz
6	31	1	Classical
7	33	1	Classical
8	37	1	Classical
9	20	0	Dance
10	21	0	Dance
11	25	0	Dance
12	26	0	Acoustic
13	27	0	Acoustic
14	30	0	Acoustic
15	31	0	Classical
16	34	0	Classical
17	35	0	Classical

In [2]:
```
# Clean the data
# For our data, it has got no duplicates, its organized, so cleaning not required.
# But split the data into input (age, gender) and output (genre)
# Split the data into Training and Test Sets

X = music_data.drop(columns = ["genre"])
X
```

Out[2]:

	age	gender
0	20	1
1	23	1
2	25	1
3	26	1
4	29	1
5	30	1
6	31	1
7	33	1
8	37	1
9	20	0
10	21	0
11	25	0
12	26	0
13	27	0
14	30	0
15	31	0
16	34	0
17	35	0

In [3]:
```
y = music_data["genre"]
y
```

Out[3]:
```
0         HipHop
1         HipHop
2         HipHop
3           Jazz
4           Jazz
5           Jazz
6      Classical
7      Classical
8      Classical
9          Dance
10         Dance
11         Dance
12      Acoustic
13      Acoustic
14      Acoustic
15     Classical
16     Classical
17     Classical
Name: genre, dtype: object
```

In [9]:
```
# General thumb rule - 80% for training, 20% for testing

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

In [18]:
```
# Create a model - Algorithms - here we use decision tree using scikit-leran library
# The more data we give it to our model and cleaner the data is, better the results we get - key concept in ML
# If we have duplicate, irrelevant, incomplete values, our model will learn bad pattern from our data.
# We should have enough data to get the accurate model.

model = DecisionTreeClassifier()

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

# Make predictions
predictions = model.predict(X_test)
predictions

# Evaluate and improve
# Compare the predictions with actual values of test we have, then we get accuracy of model.
# below function returns accuracy score between 0 - 1
# when you re-run, everytime we split into different data sets, because train_test_split this function picks data
# randomly, hence accuracy changes, for huge data set after running thousands of times and training model,
# we get some constant accuracy score.
score = accuracy_score(y_test, predictions)
score
```

Out[18]:
```
1.0
```

In [24]:
```
# You build and train the model and then we save it to a file. Now next time, we want to make predictions,
# we simply load the model from the file and ask it to make predictions. That model is already trained.
# We don't need to retrain it, it's like a intelligent person. Let us see how to do this, it's very easy, import joblib.

import pandas as pd
from sklearn.tree import DecisionTreeClassifier
import joblib
# joblib object has methods for saving and loading the models


music_data = pd.read_csv("/Users/roja/Downloads/Python Mini Projects Using Jupyter/music.csv")
X = music_data.drop(columns = ["genre"])
y = music_data["genre"]

model = DecisionTreeClassifier()
model.fit(X, y)

# Storing a trained model in a file
joblib.dump(model, 'music-recommender.joblib')
```

Out[24]:
```
['music-recommender.joblib']
```

In [26]:
```
# We don't want train our model everytime, we just have to make use of the model.
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
import joblib

# now load the file to make predictions
model = joblib.load('music-recommender.joblib')
predictions = model.predict([[21, 1]])
predictions
```

Out[26]:
```
array(['HipHop'], dtype=object)
```

In [28]:
```
# Visualizing the decision trees
# Trees in graphical format
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

music_data = pd.read_csv("/Users/roja/Downloads/Python Mini Projects Using Jupyter/music.csv")
X = music_data.drop(columns = ["genre"])
y = music_data["genre"]

model = DecisionTreeClassifier()
model.fit(X, y)

tree.export_graphviz(model, out_file = 'music-recommender.dot', feature_names = ['age', 'gender'],
                     class_names = sorted(y.unique()), label = 'all', rounded = True, filled = True )
```

In [ ]: