In [30]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn import tree
from io import StringIO
import pydotplus
from IPython.display import Image

# Load the dataset which was used throughout the whole course
data = pd.read_excel('finally_clean_data_for_plotting.xlsx')

# Show the data frame
print(data)
```

```
       age  sex  householdincome  howoftenwine  noofwines     income_category
0       34    2               12            10          1  $50,000 to $59,999
1       84    2                7             6          1  $20,000 to $24,999
2       29    2               13            10          1  $60,000 to $69,999
3       68    2                6             5          1  $15,000 to $19,999
4       54    2               11             9          1  $40,000 to $49,999
...    ...  ...              ...           ...        ...                 ...
14556   18    2                1             9          1    Less than $5,000
14557   18    1                1            10          1    Less than $5,000
14558   51    1                6             6          1  $15,000 to $19,999
14559   21    1                1            10          2    Less than $5,000
14560   18    2                1            10          1    Less than $5,000

[14561 rows x 6 columns]
```

The dataset focusses on the correlations between age, sex, household-income (already pre-categorized according to the NESARC codebook) and wine drinking frequency with the "noofwines" variable, which represents the consumed amount of wine per occasion. In the following, the data pre-processing is performed, which includes bining and grouping of the age and frequency variable, to reduce the complexity of the resulting tree. I previously tried it without the binning and grouping first, but it did not work out.

In [ ]:

```
# Data Cleaning
data_clean = data.dropna()

# Binning the age column, to reduce the complexity of the tree
bins = [0, 18, 30, 45, 60, 75, 100]
labels = ['<18', '18-30', '30-45', '45-60', '60-75', '75+']
data_clean['age_group'] = pd.cut(data_clean['age'], bins = bins, labels = labels)

# Combining categories in howoftenwine to reduce the complexity of the tree
data_clean['howoftenwine'] = data_clean['howoftenwine'].replace({
    '1 or 2 times in the last year': 'Rarely',
    '2 to 3 times a month': 'Occasionally',
    'Once a week': 'Regularly',
    '3 to 6 times in the last year': 'Rarely',
})

# Convert categorical variables to numerical
data_clean = pd.get_dummies(data_clean, columns = ['age_group', 'sex', 'howoftenwine
```

After finishing the pre-processing of the data, I proceed with the model definition. As in the course example I use a 60 to 40 train-test-split.

In [31]:

```
# Define predictors and target
predictors = data_clean.drop('noofwines', axis= 1)  # Drop the target variable
targets = data_clean['noofwines']

# Split into training and testing sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, targets, te

# Build model on training data with pruning
classifier = DecisionTreeClassifier(max_depth=3)  # Limit the depth of the tree
classifier.fit(pred_train, tar_train)

# Make predictions
predictions = classifier.predict(pred_test)

# Evaluate the model
conf_matrix = confusion_matrix(tar_test, predictions)
accuracy = accuracy_score(tar_test, predictions)

print("Confusion Matrix:\n", conf_matrix)
print("Accuracy Score:", accuracy)
```

```
Confusion Matrix:
 [[3536    0    0    0    0    0    0    0    0    0]
 [1808    0    0    0    0    0    0    0    0    0]
 [ 334    0    0    0    0    0    0    0    0    0]
 [  97    0    0    0    0    0    0    0    0    0]
 [  20    0    0    0    0    0    0    0    0    0]
 [  12    0    0    0    0    0    0    0    0    0]
 [   2    0    0    0    0    0    0    0    0    0]
 [   3    0    0    0    0    0    0    0    0    0]
 [   2    0    0    0    0    0    0    0    0    0]
 [  11    0    0    0    0    0    0    0    0    0]]
Accuracy Score: 0.6070386266094421
```

Interpretation of the results:

The accuracy score of approximately 60.7% indicates that about 60.7% of the total predictions made by the model were correct. Further, the conclusion can be drawn that the model performs well for class 0, with a high number of correct predictions of 3536, but it fails to predict any instances for classes 1 through 9, as indicated by the zeros in those rows. This suggests that the model may be biased towards class 0 or that the other classes are underrepresented.

For better understanding, the class 0 corresponds to 1 unit, class 1 to 2 units of wine, and so on.

Now the tree can be drawn:

In [32]:

```
# Create a PNG image
png_image = graph.create_png()

# Display the image
display(Image(png_image, width=1200, height=2400))
```