In [6]:
```python
from pandas import Series, DataFrame
import pandas as pd
import numpy as np
import os
import matplotlib.pylab as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
import sklearn.metrics

# Feature Importance
from sklearn import datasets
from sklearn.ensemble import ExtraTreesClassifier


# 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 [9]:
```
# 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 choose a test-train split of 60:40.

In [10]:
```
# Defining the number of wine units consumed per occasion as my target variable
targets = data_clean['noofwines']

# Specifying the predictors, excluding the "noofwines" variable
predictors = data_clean.drop(columns=['noofwines'])


pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, targets, te

pred_train.shape
pred_test.shape
tar_train.shape
tar_test.shape

#Build model on training data
from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(n_estimators = 25)
classifier = classifier.fit(pred_train,tar_train)

predictions = classifier.predict(pred_test)

sklearn.metrics.confusion_matrix(tar_test,predictions)
sklearn.metrics.accuracy_score(tar_test, predictions)


# Fit an Extra Trees model to the data
model = ExtraTreesClassifier()
model.fit(pred_train,tar_train)

# Display the relative importance of each attribute
print(model.feature_importances_)
```

```
[0.59909839 0.03423693 0.00206552 0.00729036 0.00962419 0.00781309
```

```
 0.00766901 0.00887377 0.01089343 0.01142832 0.00728565 0.00645529
 0.0084302  0.01084036 0.01062625 0.01239244 0.01024027 0.0096258
 0.01338634 0.03540826 0.00295272 0.00796803 0.00924707 0.00557615
 0.00722881 0.00581902 0.00935762 0.00667761 0.00988457 0.00557733
 0.00972883 0.00976691 0.00967726 0.01099075 0.00517176 0.01199008
 0.01095418 0.01101555 0.00527956 0.00794851 0.00783838 0.00566543]
```

As for the course example, I now run a different number of trees and see the effect of that on the accuracy of the prediction

In [11]:

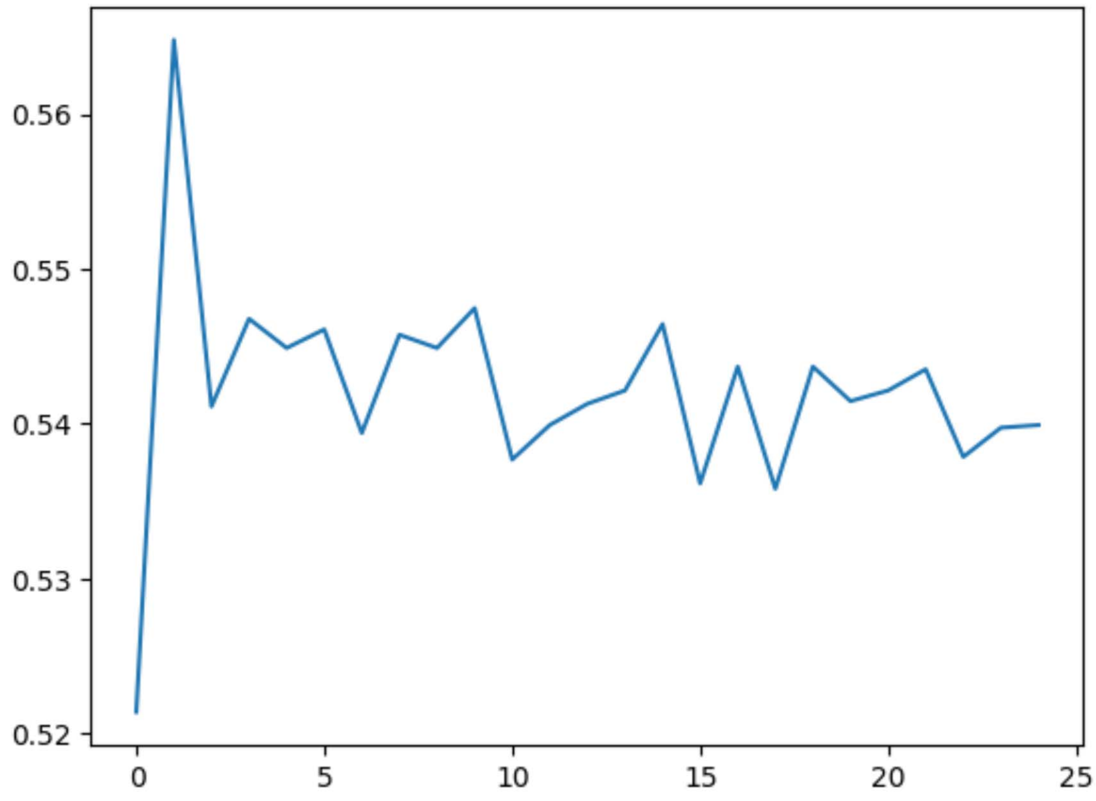
```
# Running a different number of trees and see the effect of that on the accuracy of
trees=range(25)
accuracy=np.zeros(25)

for idx in range(len(trees)):
   classifier=RandomForestClassifier(n_estimators=idx + 1)
   classifier=classifier.fit(pred_train,tar_train)
   predictions=classifier.predict(pred_test)
   accuracy[idx]=sklearn.metrics.accuracy_score(tar_test, predictions)

plt.cla()
plt.plot(trees, accuracy)
```

Out[11]:

```
[<matplotlib.lines.Line2D at 0x1ea3beca690>]
```



Interpretation of the results:

An initial increase in accuracy is observed for 1 tree with an accuracy of approximately 0.57. This indicates that with just one tree, the model is able to classify about 57% of the test samples correctly. After the initial increase, the accuracy decreases again to values between 0.54 and 0.55.

This suggests that adding more trees beyond the first one is not improving the model's performance and may even be leading to worse predictions.