I performed a Lasso Regression, straight forward following the course example, using the NESARC database and focus on wine drinking behavior.

```
In [7]:
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LassoLarsCV

# 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 [8]:
# 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 train-test split of 70:30.

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

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

# Split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, tes

# Specify the lasso regression model
model=LassoLarsCV(cv = 10, precompute = False).fit(pred_train,tar_train)
```

Now let's generate the coefficient progression, mean square error plot, as well as MSE and R-square for further analysis (just like shown in the course example code):

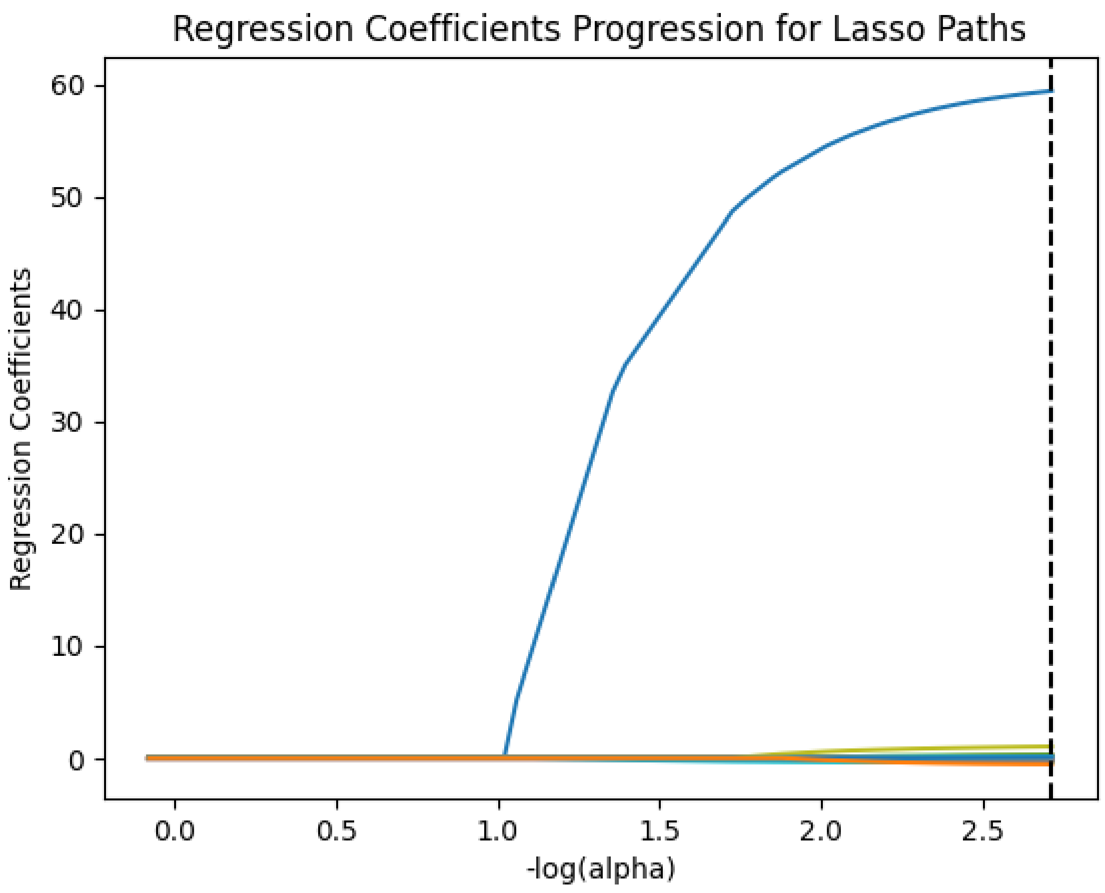
In [13]:

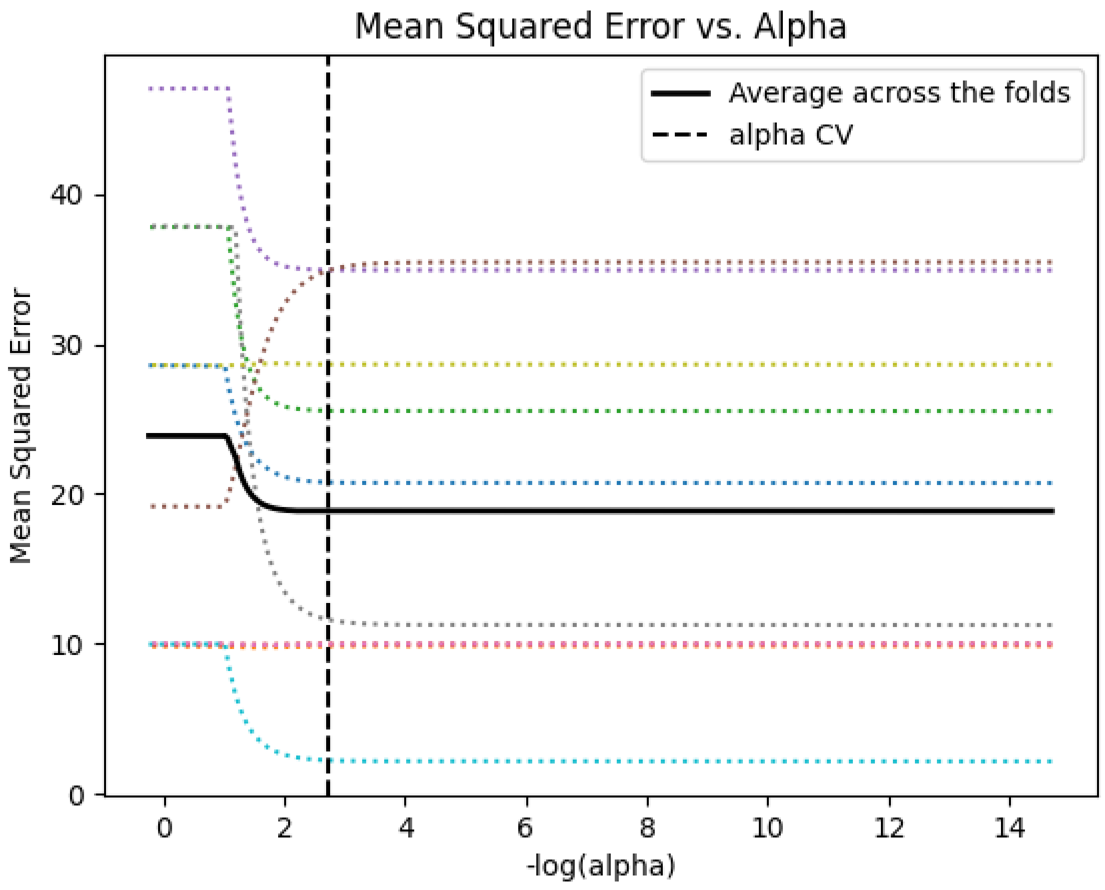
```
# Plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
plt.plot(m_log_alphas, model.coef_path_.T)
plt.axvline(-np.log10(model.alpha_), linestyle = '--', color = 'k', label = 'alpha CV
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# Plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.mse_path_, ':')  # Change cv_mse_path_ to mse_path_
plt.plot(m_log_alphascv, model.mse_path_.mean(axis = -1), 'k', label = 'Average acros
plt.axvline(-np.log10(model.alpha_), linestyle = '--', color = 'k', label = 'alpha CV
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean Squared Error')
plt.title('Mean Squared Error vs. Alpha')
plt.show()

# MSE from training and test data
from sklearn.metrics import mean_squared_error
train_error = mean_squared_error(tar_train, model.predict(pred_train))
test_error = mean_squared_error(tar_test, model.predict(pred_test))
print ('training data MSE')
print(train_error)
print()
print ('test data MSE')
print(test_error)
print()

# R-square from training and test data
rsquared_train = model.score(pred_train,tar_train)
rsquared_test = model.score(pred_test,tar_test)
print ('training data R-square')
print(rsquared_train)
print()
print ('test data R-square')
print(rsquared_test)
```





```
training data MSE
17.96558436929015
```

```
test data MSE
7.899428563059829

training data R-square
0.24747672832188616

test data R-square
0.42056956171672644
```

Interpretation of results:

Regression Coefficients Progression:

- The lines that are close to zero indicate that those features have very little influence on the target variable when the alpha is applied. Hence, the features are less important and have been effectively excluded, suggesting they may not be useful for predicting the target variable.
- The blue line that starts that increases to a regression coefficient of 60 indicates that this particular feature has a significant positive relationship with the target variable. As alpha decreases, the regularization effect is reduced, allowing this feature's coefficient to increase. This suggests that this feature becomes more influential in the model as regularization is relaxed

MSE-Plot:

The average value across all folds is 25, reducing to 20 which indicates that, the squared difference between the predicted number of drinks and the actual number of drinks is 25 or 20. This means that, on average, the model's predictions are off by about 5 or 4.47 drinks (square root of 25 or 20, respectively).

Mean Squared Error (MSE)

Training Data MSE: 17.97 This value indicates that, on average, the squared difference between the predicted number of drinks and the actual number of drinks in the training set is approximately 17.97 (RMS error = 4.24) as already roughly estimated, based on the plot.

Test Data MSE: 7.90 This value indicates that the average squared difference between the predicted and actual values in the test set is approximately 7.90 (RMS error = 2.81). This is a better performance compared to the training set, suggesting that the model generalizes better to data to be analyzed than it does to the training data.

Since the target variable only ranges from 1 to 12 (drinks per occasion), the above suggests that the model is overall not performing well.

R-squared:

Training Data R^2: 0.247 The value can be interpreted as approximately 24.7% of the variance in the training data can be explained by the model. This is relatively low and also indicates a bad performance, since the model may be underfitting.

Test Data R^2: 0.421 As for the training data, the value can be interpreted as about 42.1% of the variance in the test data can be explained by the model. This is significantly better than for the training data and indicates, that the model generalizes better to data which is to be analyzed (unseen yet). However, it is still relatively low, indicating that there is still a significant amount of unexplained variance in the test data.