Question1.1

Here is the code

```
import pandas as pd
import numpy as np
from sklearn.cluster import AgglomerativeClustering
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data"
columns = ['mpg', 'cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', 'model_year', 'origin', 'car_name']
df = pd.read_csv(url, sep='\s+', names=columns)  # Use sep='\s+' instead of delim_whitespace

continuous_cols = ['mpg', 'cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', 'model_year']

df['horsepower'] = pd.to_numeric(df['horsepower'], errors='coerce')

X = df[continuous_cols].copy()

imputer = SimpleImputer(strategy='mean')
X_imputed = imputer.fit_transform(X)
X_imputed = pd.DataFrame(X_imputed, columns=continuous_cols)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_imputed)

clustering = AgglomerativeClustering(n_clusters=3, linkage='average', metric='euclidean')  # Use metric instead of affinit
labels = clustering.fit_predict(X_scaled)

df['cluster'] = labels

cluster_stats = df.groupby('cluster')[continuous_cols].agg(['mean', 'var'])
print("Cluster Statistics (Mean and Variance):")
print(cluster_stats)

origin_stats = df.groupby('origin')[continuous_cols].agg(['mean', 'var'])
print("\nOrigin Class Statistics (Mean and Variance):")
print(origin_stats)

crosstab = pd.crosstab(df['cluster'], df['origin'])
print("\nCrosstab of Cluster vs Origin:")
print(crosstab)

print("\nAnalysis:")
if crosstab.max().max() / crosstab.sum().sum() > 0.5:
    print("There is a clear relationship between cluster assignments and origin labels.")
else:
    print("There is no clear relationship between cluster assignments and origin labels.")
```

Here is the output

```
Cluster Statistics (Mean and Variance):
               mpg                cylinders              displacement               \
              mean          var        mean        var          mean          var
cluster
0        26.214576   41.397984   4.620339   0.998224    143.391525  3416.251799
1        14.653535    5.377819   8.000000   0.000000    346.626263  2126.705834
2        43.700000    0.300000   4.000000   0.000000     91.750000    12.250000

         horsepower                    weight                   acceleration  \
               mean          var         mean             var           mean
cluster
0         86.093426   308.994714  2593.162712  296800.095884     16.433220
1        160.353535   726.679860  4128.393939  202509.139147     12.694949
2         49.000000     4.000000  2133.750000   21672.916667     22.875000

                    model_year
              var         mean        var
cluster
0        4.900525   76.732203  13.121919
1        3.263750   73.696970   8.315399
2        2.309167   80.000000   2.666667

Origin Class Statistics (Mean and Variance):
              mpg               cylinders              displacement               \
             mean         var        mean        var          mean          var
origin
1       20.083534   40.997026   6.248996   2.760332    245.901606  9702.612255
2       27.891429   45.211230   4.157143   0.250311    109.142857   509.950311
3       30.450633   37.088685   4.101266   0.348588    102.708861   535.465433

        horsepower                      weight                  acceleration  \
              mean          var         mean             var           mean
origin
1       119.048980  1591.833657  3361.931727  631695.128385     15.033735
2        80.558824   406.339772  2423.300000  240142.328986     16.787143
3        79.835443   317.523856  2221.227848  102718.485881     16.172152

                   model_year
             var         mean        var
origin
1       7.568615   75.610442  13.521020
2       9.276209   75.814286  12.037474
3       3.821779   77.443038  13.326842

Crosstab of Cluster vs Origin:
origin      1    2    3
cluster
0         150   66   79
1          99    0    0
2           0    4    0

Analysis:
There is no clear relationship between cluster assignments and origin labels.
```

From the image, we can see that Cluster 1 and Cluster 2 have a strong association with the origin categories, while Cluster 0 is a mixed category primarily containing American and Japanese cars but with features closer to European cars. The hierarchical clustering did not fully distinguish between origin=1 (American) and origin=3 (Japanese) vehicles.

Therefore, my conclusion is:

There is a partial association between cluster assignments and class labels, but no particularly strong or clear relationship.

Question1.2

Here is my code

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score

try:
    from sklearn.datasets import load_boston
    boston = load_boston()
    X = pd.DataFrame(boston.data, columns=boston.feature_names)
    dataset_name = "Boston"
except ImportError:
    from sklearn.datasets import fetch_california_housing
    boston = fetch_california_housing()
    X = pd.DataFrame(boston.data, columns=boston.feature_names)
    dataset_name = "California Housing"

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

silhouette_scores = []
for k in range(2, 7):
    kmeans = KMeans(n_clusters=k, n_init=10, random_state=42)  # Explicitly set n_init
    labels = kmeans.fit_predict(X_scaled)
    score = silhouette_score(X_scaled, labels, sample_size=1000, random_state=42)  # Limit sample size
    silhouette_scores.append((k, score))

optimal_k = max(silhouette_scores, key=lambda x: x[1])[0]
print(f"Optimal number of clusters for {dataset_name}: {optimal_k}")
print(f"Silhouette scores for k=2 to 6: {silhouette_scores}")

kmeans_optimal = KMeans(n_clusters=optimal_k, n_init=10, random_state=42)
labels_optimal = kmeans_optimal.fit_predict(X_scaled)

X['cluster'] = labels_optimal

cluster_means = X.groupby('cluster').mean()
print(f"\nMean values for each cluster in {dataset_name}:")
print(cluster_means)

centroids = pd.DataFrame(scaler.inverse_transform(kmeans_optimal.cluster_centers_), columns=boston.feature_names)
print(f"\nCentroid coordinates for {dataset_name}:")
print(centroids)

print(f"\nDifference between cluster means and centroids in {dataset_name}:")
print(cluster_means - centroids)
```

Here is the output

```
Optimal number of clusters for California Housing: 2
Silhouette scores for k=2 to 6: [(2, 0.3368433874602563), (3, 0.3368433874602563), (4, 0.3026463930769773), (5, 0.32344236329026266), (6, 0.3232918404758
393)]

Mean values for each cluster in California Housing:
           MedInc   HouseAge  AveRooms  AveBedrms   Population  AveOccup  \
cluster
0        3.918104  28.412773  5.225159   1.075685  1532.241745  3.098100
1        3.805276  28.952057  5.710036   1.125614  1278.279590  3.032817

          Latitude   Longitude
cluster
0        33.945736 -118.010096
1        37.956526 -121.719940

Centroid coordinates for California Housing:
     MedInc   HouseAge  AveRooms  AveBedrms   Population  AveOccup   Latitude  \
0  3.918418  28.414681  5.225372   1.075692  1532.198896  3.097956  33.945450
1  3.804868  28.949303  5.709630   1.125593  1278.397166  3.033030  37.955996

    Longitude
0 -118.009704
1 -121.719626

Difference between cluster means and centroids in California Housing:
           MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  \
cluster
0       -0.000315 -0.001908 -0.000213  -0.000007    0.042849  0.000144
1        0.000407  0.002754  0.000406   0.000021   -0.117576 -0.000213

         Latitude  Longitude
cluster
0        0.000286  -0.000392
1        0.000530  -0.000315
```

Based on the output results, it can be concluded that the optimal k is 2, with the two clusters primarily reflecting differences in housing characteristics between the northern and southern regions of California.

Cluster 0 is concentrated in the southern region,

Cluster 1 is concentrated in the northern region.

The differences between the mean values and centroid coordinates are very minor, indicating that the K-Means clustering results are reliable. The mean values and centroid coordinates are almost numerically idendtical.

Question1.3

Here is my code

```
import os
import pandas as pd
import numpy as np
from sklearn.datasets import load_wine
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import homogeneity_score, completeness_score

os.environ["OMP_NUM_THREADS"] = "1"

wine = load_wine()
X = pd.DataFrame(wine.data, columns=wine.feature_names)
y = wine.target

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

kmeans = KMeans(n_clusters=3, n_init=10, random_state=42)  # Explicitly set n_init
labels = kmeans.fit_predict(X_scaled)

homogeneity = homogeneity_score(y, labels)
completeness = completeness_score(y, labels)

print("Homogeneity Score:", homogeneity)
print("Completeness Score:", completeness)
```

Here is the output

```
Homogeneity Score: 0.8788432003662366
Completeness Score: 0.8729636016078731
```

The homogeneity and completeness scores are 0.878 and 0.873, respectively, indicating that the K-Means clustering performs well on the Wine dataset.

Using homogeneity and completeness together provides a comprehensive evaluation of how well the clustering results align with the true labels. The current high scores demonstrate that K-Means effectively captures the class structure of the Wine dataset.