```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.mixture import GaussianMixture

data_file = 'data_ex1_wt.csv' # absolute path respect to when the script is run
df = pd.read_csv(data_file, header=None, names=['time', 'metric'])

x = df['time'].values
y = df['metric'].values

#### REMOVING TREND FROM THE DATA
m = 5 #best degree

def least_squares_methods(x, y, degree):

    #compute Vandermonde matrix
    A = []

    for i in range(len(x)):
        A.append([])

    for j in range(len(x)):
        for i in range(degree+1):
            A[j].append(x[j]**i)

    #transpose of A
    A_transpose = np.transpose(A)

    # Compute A' A
    A_transpose_A = np.dot(A_transpose, A)
    # Compute (A' A)^-1
    A_transpose_A_inv = np.linalg.inv(A_transpose_A)
    # Compute (A' A)^-1 A'
    A_transpose_A_inv_transpose_A = np.dot(A_transpose_A_inv, A_transpose)
    # Compute (A' A)^-1 A' y (= b)
    b = np.dot(A_transpose_A_inv_transpose_A, y)

    return b

p = least_squares_methods(x, y, m)
yy = np.zeros(len(x))
for j in range(len(p)):
    yy += p[j] * (x**j)

residuals = y - yy
y = residuals

###Try to find the optimal number of variances

k_values = range(1, 6)
n_init = 10
#we perform 10 times the algorithm, every time with different initialization of the parameters
#then it keeps the initialization that generates highest likelihood (it guarantees more robust results)

aic, bic, log_likelihoods = [], [], []

for k in k_values:
    gmm = GaussianMixture(n_components=k, n_init=n_init)
    gmm.fit(y.reshape(-1, 1))

    log_likelihood_total = gmm.score(y.reshape(-1, 1)) * len(y)
    aic_k = gmm.aic(y.reshape(-1, 1))
    bic_k = gmm.bic(y.reshape(-1, 1))

    log_likelihoods.append(log_likelihood_total)
    aic.append(aic_k)
    bic.append(bic_k)
    print(f"AIC with {k}: {aic_k}")
    print(f"BIC with {k}: {bic_k}")
    print(f"Log-likelihood with {k}: {log_likelihood_total}")

plt.figure(figsize=(10, 5))
plt.plot(k_values, aic, marker='o', label='AIC')
plt.plot(k_values, bic, marker='o', label='BIC')
#plt.plot(k_values, log_likelihoods, marker='o', label='Log-Likelihood') - also the log-likelihood of the dataset is a meaningful metric
plt.xlabel('Number of Gaussians (k)')
plt.ylabel('Criterion Value')
plt.title('AIC/BIC vs. Number of Gaussians')
plt.xticks(k_values)
plt.legend()
plt.grid(True)
plt.show()
```