```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

##### DATA PREPARATION
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

##### LEAST SQUARES POLYNOMIAL FITTING with plotting

errors = [] #mean squared errors
M = 10 #max degree of polynomial

plt.figure(figsize=(8, 4))

#colormap for better line distinction
colors = [
    "#e6194b",
    "#3cb44b",
    "#ffe119",
    "#0082c8",
    "#f58231",
    "#911eb4",
    "#46f0f0",
    "#f032e6",
    "#d2f53c",
    "#fabebe"
]

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

for i in range(1, M+1):
    lib_coeffs = np.polyfit(x, y, i)
    coeffs = least_squares_methods(x, y, i)

    #print the coefficients of the polynomial
    print(f"Degree {i} Coefficients:", coeffs)

    #just to verify the correctness
```

```
    print(f"Degree {i} Coefficients with library method:", lib_coeffs)

    #calculate the polynomial values
    yy = np.zeros(len(x))
    for j in range(len(coeffs)):
        yy += coeffs[j] * (x**j)

    errors.append(np.mean((y - yy)**2))

    plt.plot(x, yy, label=f'Degree {i}', linewidth=2, color=colors[i - 1])

plt.scatter(x, y, color='gray', alpha=0.5)

plt.title("Time vs Metric", fontsize=14)
plt.xlabel("Time", fontsize=12)
plt.ylabel("Metric", fontsize=12)

plt.tick_params(axis='x', rotation=45)
plt.ylim(min(y)-1, max(y)+1)
plt.legend(title="Polynomial Degrees")
plt.tight_layout()
plt.show()

#We can graphically see that any polymonial of degree higher than 5 is enough to fit the data

for i in range(len(errors)):
    print(f"Mean Squared Error for degree {i+1}: {errors[i]}")

### FIND THE OPTIMAL DEGREE
plt.figure(figsize=(8, 4))
plt.plot(range(1, M+1), errors, marker='o', linestyle='-', color='darkred', linewidth=2)
plt.title("MSE vs Polynomial Degree", fontsize=14)
plt.xlabel("Polynomial Degree", fontsize=12)
plt.ylabel("Mean Squared Error", fontsize=12)
plt.xticks(range(1, M+1))

for i, error in enumerate(errors, start=1):
    plt.text(i+0.2, error + 0.2, f'{error:.4f}', ha='center', fontsize=10)


#We can see it is 5 since after that the error is not decreasing significantly
plt.tight_layout()
plt.show()
```