Task 1:

```
# -*- coding: utf-8 -*-
"""Alaiba_Nawaz_Day3_W2.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1KYiXzWnSAEyr1JRV0f9zEzfaDweCZF7x
"""

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import scipy.stats as stats

data = pd.read_csv("/content/heart.csv")

data.info()

#checking for nulls
data.isnull().any()

#hist for chol
plt.hist(data['chol'], edgecolor='black')
plt.xlabel('age')
plt.ylabel('Count')
plt.title('Distribution of chol')
plt.show()

#hist for target
plt.hist(data['target'], edgecolor='black')
plt.xlabel('target')
plt.ylabel('Count')
plt.title('Distribution of target')
plt.show()

#boxplot for chol to see outliers
plt.boxplot(data['chol'])
```

```
plt.show()

#removing the outliers
#printing numbers instances before removal of outliers
print(data["chol"].value_counts())
# Calculate the first quartile (Q1) and third quartile (Q3)
Q1 = np.percentile(data['chol'], 25)
Q3 = np.percentile(data['chol'], 75)
# Calculate the interquartile range (IQR)
IQR = Q3 - Q1
# Define the upper and lower bounds for outlier detection
lower_bound = Q1 - 1 * IQR
upper_bound = Q3 + 1 * IQR
# Remove outliers from the DataFrame
data = data[(data['chol'] >= lower_bound) & (data['chol'] <= upper_bound)]
#printing numbers instances after removal of outliers
print(data["chol"].value_counts())

correlation = data['target'].corr(data['chol'])
print(correlation)

"""Null Hypothesis (H₀): There is no significant difference in cholesterol
levels between patients with and without heart disease.

Alternate Hypothesis (H₁): There is a significant difference in
cholesterol levels between patients with and without heart disease.
"""

#separating patients with disease and with no disease
no_heart_disease = data[(data["target"]==0)]
with_heart_disease = data[(data["target"]==1)]

# Perform independent t-test
t_statistic, p_value = stats.ttest_ind(with_heart_disease["chol"],
no_heart_disease["chol"])

# Print the test results
print("Independent t-test results:")
print("t-statistic:", t_statistic)
print("p-value:", p_value)
```

```
# Calculate confidence intervals
confidence_interval_no_disease = stats.t.interval(0.95,
len(no_heart_disease["chol"]) - 1,

loc=no_heart_disease["chol"].mean(),

scale=stats.sem(no_heart_disease["chol"]))
confidence_interval_disease = stats.t.interval(0.95,
len(with_heart_disease["chol"]) - 1,

loc=with_heart_disease["chol"].mean(),

scale=stats.sem(with_heart_disease["chol"]))

# Print the confidence intervals
print("Confidence interval (no heart disease):",
confidence_interval_no_disease)
print("Confidence interval (with heart disease):",
confidence_interval_disease)
alpha = 0.05

if p_value < alpha:
    print("Reject the null hypothesis.")
else:
    print("Fail to reject the null hypothesis.")

# Create a box plot
plt.boxplot([no_heart_disease["chol"], with_heart_disease["chol"]])
plt.xticks([1, 2], ['No Heart Disease', 'With Heart Disease'])
plt.ylabel("Cholesterol Levels")
plt.title("Comparison of Cholesterol Levels")
plt.show()

# Create histograms or density plots
sns.histplot(no_heart_disease["chol"], label="No Heart Disease")
sns.histplot(with_heart_disease["chol"], label="With Heart Disease")
plt.xlabel("Cholesterol Levels")
plt.ylabel("Density")
plt.legend()
```

```
plt.show()

"""We reject the null hypothesis as the p value is less tha alpha which
means that there is a significant difference in cholesterol levels between
patients with and without heart disease.
```

Task 2:

```
Task 2
"""

import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from PIL import Image

def kmeans_image(image_path, k):
    # Load the image
    image = Image.open(image_path)
    rgb_image = image.convert("RGB")

    # Convert image to numpy array
    image_array = np.array(rgb_image)
    h, w, _ = image_array.shape

    # Reshape the array to fit the KMeans algorithm
    reshaped_array = image_array.reshape(h * w, -1)

    # Perform k-means clustering
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(reshaped_array)

    # Get the labels and cluster centers
    labels = kmeans.predict(reshaped_array)
    centers = kmeans.cluster_centers_

    # Create a new image with the cluster colors
    new_image_array = np.zeros_like(reshaped_array)
    for i in range(k):
        new_image_array[labels == i] = centers[i]
```

```
    # Reshape the new image array
    new_image_array = new_image_array.reshape(h, w, -1)

    # Convert the array back to image
    new_image = Image.fromarray(new_image_array.astype(np.uint8))

    # Plot the original and clustered images
    fig, axes = plt.subplots(1, 2, figsize=(10, 5))
    axes[0].imshow(image)
    axes[0].set_title("Original Image")
    axes[0].axis("off")
    axes[1].imshow(new_image)
    axes[1].set_title(f"K-Means Clustering (K={k})")
    axes[1].axis("off")
    plt.show()

# Define the image path
image_path = "path.jpg"

# Perform k-means clustering for different values of k
k_values = [2, 3, 5, 10, 15, 20]
for k in k_values:
    kmeans_image("/content/face.jpg", k)
```

Task 3:

This tutorial teaches us how to use Amazon SageMaker, a service that helps you build, train, and deploy machine learning models. We focus on using the XGBoost algorithm to create a model that predicts if a customer will sign up for a bank's certificate of deposit (CD). This guide is designed for developers and data scientists who want to use SageMaker to make the process of building and deploying ML models easier.

These were the steps:

1. Create a SageMaker notebook instance
2. Prepare the data
3. Train the model to learn from the data
4. Deploy the model
5. Evaluate your ML model's performance

Amazon SageMaker makes it easier to create, train, and deploy machine learning models from start to finish. It takes away the hassle of managing the technical infrastructure, so developers can concentrate on building and deploying their models. SageMaker can handle large amounts

of data and scale up as needed. It's also cost-effective, and the resources used in this tutorial are available for free under the AWS Free Tier.

Conclusion :
This tutorial showed how to use Amazon SageMaker to build, train, and deploy ML models using the XGBoost algorithm. SageMaker's managed environment simplifies model development and deployment for developers and data scientists. With scalability and easy infrastructure management, SageMaker is an effective tool for real-world ML model building and deployment