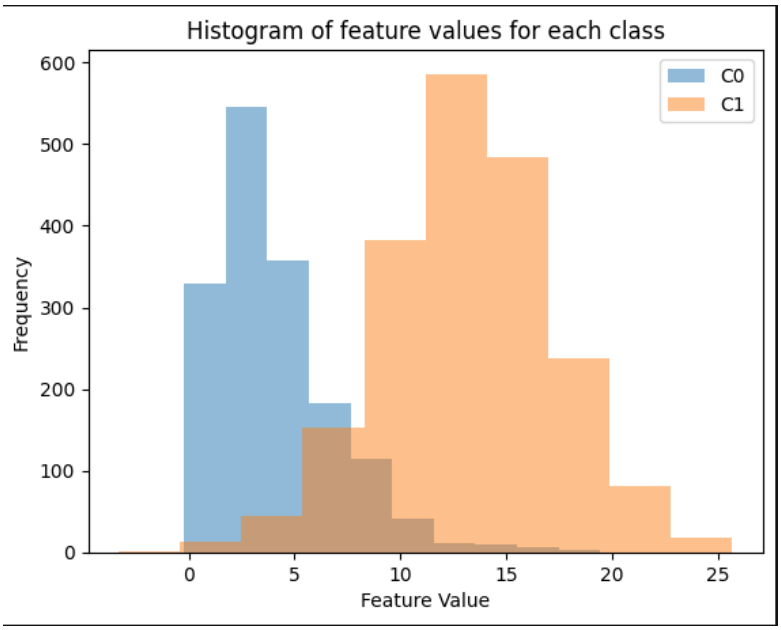
FYS-2021 mandatory assignment

Problem 1

A) As seen in the histogram, we have data points that have a value that can make classification difficult. Especially data points that have a value between 5 and 10 will be harder to classify correctly with the Bayesian method. Values lower or higher than this will be easier to classify, but as seen in the histogram there will be some misclassifications. As seen from pandas.info we have 3600 data points with 2 columns as we have 1 feature and 1 class column. Using pandas.describe we can see that mean of the class column show what the histogram already show. This is that we do not have an equal amount of each class.




```
<class 'pandas.core.frame.DataFrame'>
Index: 3600 entries, 0 to 3599
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   feature  3600 non-null   float64
 1   class    3600 non-null   float64
dtypes: float64(2)
memory usage: 84.4 KB
None (3600, 2)
           feature        class
count  3600.000000  3600.000000
mean      9.118580     0.555556
std       5.747397     0.496973
min      -3.310259     0.000000
25%       3.710522     0.000000
50%       9.091724     1.000000
75%      13.729211     1.000000
max      25.673673     1.000000
```

B) Split the data and create 2 numpy array that we can use to show MLE for Beta_hat, my_hat and sigma^2_hat. The values we get from this are are 2.03 for Beta_hat, 13.16 for my_hat and 16.17 for sigma^2_hat. This was done using pandas.loc.

C) Using pandas.replace, pandas.drop and pandas.to_numpy to create the data splits. Also creating Gamma function, gaussian function and classify function that will be used in a while loop to create the prediction model.

```
#1C

# Splitting up the data and randomizes it according to seed "random_state".
data_split = data.sample(frac = 0.8, random_state = 1)
feature_training = data_split.drop(columns = "class")
feature_test = data.drop(data_split.index)
class_training = data_split.drop(columns = "feature")
class_test = feature_test.drop(columns = "feature")
feature_test = feature_test.drop(columns = "class")


# Convert data to numpy arrays
feature_training = feature_training.to_numpy()
feature_test = feature_test.to_numpy()
class_training = class_training.to_numpy()
class_test = class_test.to_numpy()

# Create functions for classification

def Gamma(x, a, B_hat):
    return (B_hat**a) * (x**(a-1)) * np.exp(-B_hat*x) / np.math.gamma(a)

def gaussian(x, mu, sigma_squared):
    return np.exp(- (x - mu)**2 / (2 * sigma_squared)) / np.sqrt(2 * np.pi * sigma_squared)

def classify(x, a, B_hat, my_hat, sigma_squared):
    prob_C0 = Gamma(x, a, B_hat) * 0.5
    prob_C1 = gaussian(x, my_hat, sigma_squared) * 0.5

    return 0 if prob_C0 > prob_C1 else 1
```

```
# Setting variables and list for the for loop
prediction = []
index = 0

# While loop for iteration through data set
while index < len(feature_training):
    x_val = feature_training[index]

    predicted_class = classify(x_val, a, B_hat, my_hat, sigma_squared)
    prediction.append(predicted_class)
    index += 1

accuracy = np.sum((prediction == class_training) / len(class_training))
print(f"Training accuracy: {round(accuracy,2)}%.")
```

```
/tmp/ipykernel_31178/3340743229.py:23: DeprecationWarning: `np.math` is a deprecated alias fo
25). Replace usages of `np.math` with `math`
  return (B_hat**a) * (x**(a-1)) * np.exp(-B_hat*x) / np.math.gamma(a)
Training accuracy: 1516.92%.
```

I did not get enough time to get to the bottom of why my accuracy is off.