Problem 1

Berkay BARLAS
MTK 03734294
Introduction to ML

1.1) To begin with if $L(y,t)$ is convex in t, sum of $L(y,t)$ function will be convex too.

$$\frac{1}{n}\sum_{i=1}^{n} L(y_i, \langle w, x_i \rangle + b) \qquad \theta = \begin{bmatrix} b \\ w \end{bmatrix}$$

↳ unique

The loss function depends on b and w and the given function is also convex since it's sum of loss functions

1,2) Since Loss function $L(y,t)$ is convex, so $\log(1+\exp(-yt))$ is also convex.

According to lecture notes the expression is propotional to negative log-likelihood for logistic regression.

$$g(z) = (1+e^{-z})^{-1} \rightarrow \text{negative log} \rightarrow -\log(1+e^{-z})^{-1}$$

$$= \log(1+e^{-z}) = \log(1+e^{-yt})$$

$$t = \langle w, x_i \rangle + b$$

Problem 2

2.1) Prepared data and fit the Naive Bayes model

Test error changed around 20% because we are shuffling the test and training data

```
Error rate for test data: % 19
Number of mislabeled classes out of a total 2601 points : 513
```

```python
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats
from scipy.optimize import minimize
from sklearn.naive_bayes import GaussianNB

import math

# Load Data
spambase = np.loadtxt('spambase.data', delimiter=',')

# Shuffle the data,
np.random.shuffle(spambase)

# Quantize each feature variable to one of two values,
# say 0 and 1, so that values below the median map to 0, and those above map to 1.

quantize_spambase = np.copy(spambase)
rows, columns = spambase.shape

n , d = spambase.shape
train_data_size = 2000
test_data_size = n - 2000
# Calculate medians only with training data
col_medians = np.median(spambase[:2000,:], axis=0)

for i in range(rows):
    for j in range(columns-1):
        if(col_medians[j] < spambase[i][j]):
            quantize_spambase[i][j] = 1
        else:
            quantize_spambase[i][j] = 0

X = quantize_spambase[:, :-1]
y = quantize_spambase[:, -1].astype(int)


# First 2000 examples as training
train_x = X[:train_data_size,:]
train_y = y[:train_data_size]

# Rest is test data
test_x = X[train_data_size:,:]
test_y = y[train_data_size:]
```

```python
#### Part 1 ####

# Fit the Naive Bayes model using the training data
gnb = GaussianNB()
y_pred = gnb.fit(train_x, train_y).predict(test_x)

# Compute the misclassification rate (i.e., the test error) on the test data.
pred_error = (test_y != y_pred).sum()
# Report the test error.
print("Error rate for test data: %% %d" % (pred_error / test_data_size * 100))
print("Number of mislabeled classes out of a total %d points : %d" % (test_data_size, pred_error))
```

2.2) Training class majority was 0.
The error percentage came around 40%
which is twice of Naive Bayes Model.

```
Training class majority:  0
Sanity error percentage % 40.099961553248754
Number of mislabeled classes out of a total 2601 points : 513
```

```python
#### Part 2 ####

# The test error if predict the same class, namely, the majority class from the training data?
# Classes are 0 or 1
training_class_sum = np.sum(train_y)
training_class_majority = 0
if training_class_sum > (train_data_size / 2):
    #Means Majority classes are 1, else stays 0
    training_class_majority = 1

sanity_error = np.sum(np.abs(np.subtract(test_y, training_class_majority)))
sanity_error_per = np.sum(np.abs(np.subtract(test_y, training_class_majority))) / (test_data_size) * 100
print("Training class majority: ", training_class_majority, "\nSanity error percentage %" ,sanity_error_per)
print("Number of mislabeled classes out of a total %d points : %d" % (test_data_size, pred_error))
```