1a) Implementing Logitboost

As always start with importing what we need and finding our data locations.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, roc_curve, auc

import os

os.chdir("C:/Users/rique/Downloads/datasets")
```

For part a) We're using the Gisette data

load the data.

```
def load_dataset(X, y, Xtest, yTest, probChar):

    if(probChar == 'a'):
        X = np.loadtxt("gisette_train.data")
        y = np.loadtxt("gisette_train.labels")
        Xtest = np.loadtxt("gisette_valid.data")
        yTest = np.loadtxt("gisette_valid.labels")
    if(probChar == 'b'):
        X = np.genfromtxt('dexter_train.csv', delimiter=',')
        y = np.loadtxt('dexter_train.labels')
        Xtest = np.genfromtxt('dexter_valid.csv', delimiter=',')
        yTest = np.loadtxt('dexter_valid.labels')
    if(probChar == 'c'):
        X = np.loadtxt("madelon_train.data")
        y = np.loadtxt("madelon_train.labels")
        Xtest = np.loadtxt("madelon_valid.data")
        yTest = np.loadtxt("madelon_valid.labels")

    return X, y, Xtest, yTest
```

Loss

```
def logit_loss(y, h):
    return np.log(1+np.exp(-2*y*h))
```

Logitboost

```
def Logitboost(X, y, Xtest, yTest, M, k_features, iterations):

    for i in range(len(k_features)):
        x=X
        xt=Xtest
        beta = np.zeros(M)
        for j in range(0,k_features[i]):
            #print("J = ", j)
            h = X@beta

            #logitboost algo slide 28 boosting
            px = 1.0/(1.0+np.exp(-2*h))
            w = px*(1.0-px)
            z = 0.5*(y+1)-px
            z[w==0] = 0
            z[w!=0] /= w[w!=0]

            coefficients = np.zeros((2,M-1))
            losses = np.zeros(M-1)

            #to find h(x, thetak) use simple linear regression slide 9
in regression
            for k in range(0,M-1):
                 xk = x[:,k+1]
                 a=np.sum(w)
                 b=np.sum(w*xk)
                 c=np.sum(w*xk**2)
                 d=np.sum(w*z)
                 e=np.sum(w*xk*z)

                 if(a*c-b**2)==0:
                    betak=np.array([d/a,0])
                 else:
                    betak=np.array([c*d-b*e,a*e-b*d])/(a*c-b**2)

                 hk=h+0.5*(betak[0]+betak[1]*xk)
                 lossk=np.sum(logit_loss(y, hk))
                 #print("lossj", lossk)
                 coefficients[:,k]=betak
                 losses[k]=lossk

            min=np.argmin(losses)
            beta[0] += 0.5*coefficients[0,min]
            beta[min+1] += 0.5*coefficients[1][min]

            if(k_features[i] == 300):
                train_loss_300.append(losses[min])
```

```
        y_pred = np.dot(X, beta)
        yTest_pred = np.dot(Xtest, beta)
        y_pred_binary = np.where(y_pred > 0, 1, -1)
        yTest_pred_binary = np.where(yTest_pred > 0, 1, -1)

        misclass_error_train = 1 - accuracy_score(y, y_pred_binary)
        misclass_error_valid = 1 -
accuracy_score(yTest,yTest_pred_binary)

        train_misclass_errors.append(misclass_error_train)
        valid_misclass_errors.append(misclass_error_valid)

        if(k_features[i] == 100):
            # Calculate ROC curve values for the training set
            fpr_train, tpr_train, _ = roc_curve(y, 1 / (1 +
            np.exp(-y_pred)))
            roc_auc_train = auc(fpr_train, tpr_train)
            fpr_train_list.append(fpr_train)
            tpr_train_list.append(tpr_train)
            roc_auc_train_list.append(roc_auc_train)

            # Calculate ROC curve values for the validation set
            fpr_valid, tpr_valid, _ = roc_curve(yTest, 1 / (1 +
            np.exp(-yTest_pred)))
            roc_auc_valid = auc(fpr_valid, tpr_valid)
            fpr_valid_list.append(fpr_valid)
            tpr_valid_list.append(tpr_valid)
            roc_auc_valid_list.append(roc_auc_valid)

X = np.empty(1)
y = np.empty(1)
Xtest = np.empty(1)
yTest = np.empty(1)
probChar = 'a'

X, y, Xtest, yTest = load_dataset(X, y, Xtest, yTest, probChar)
```

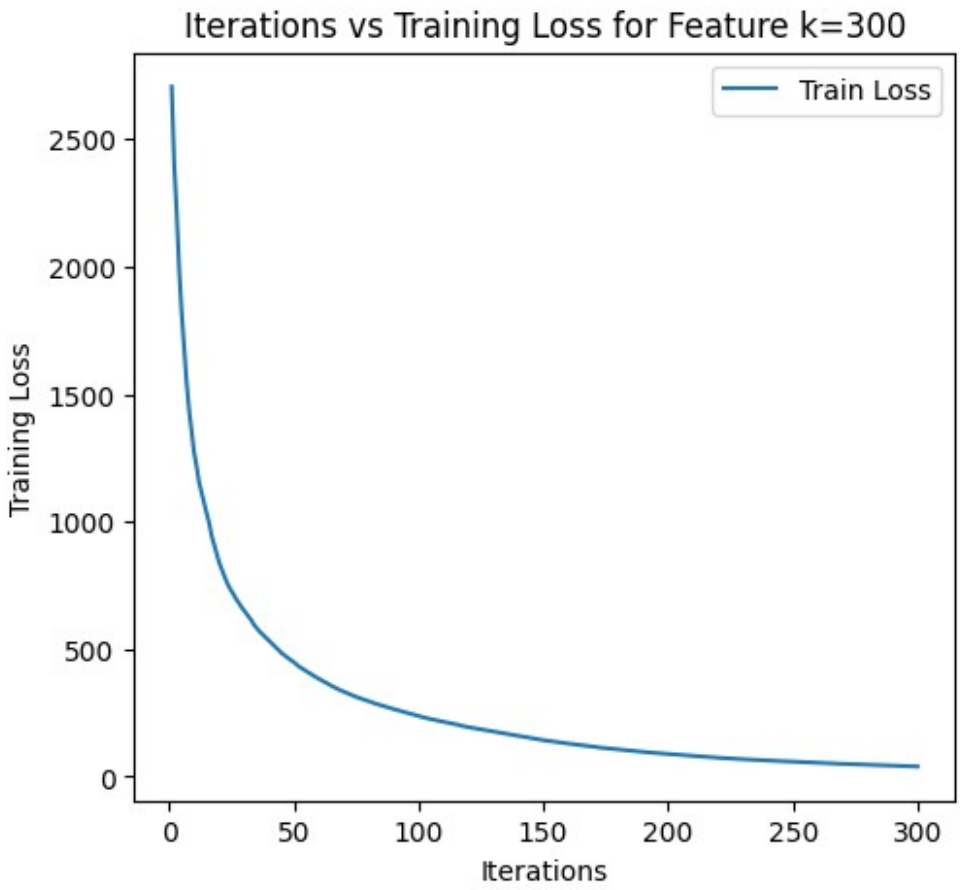
Initalize our needed parameters

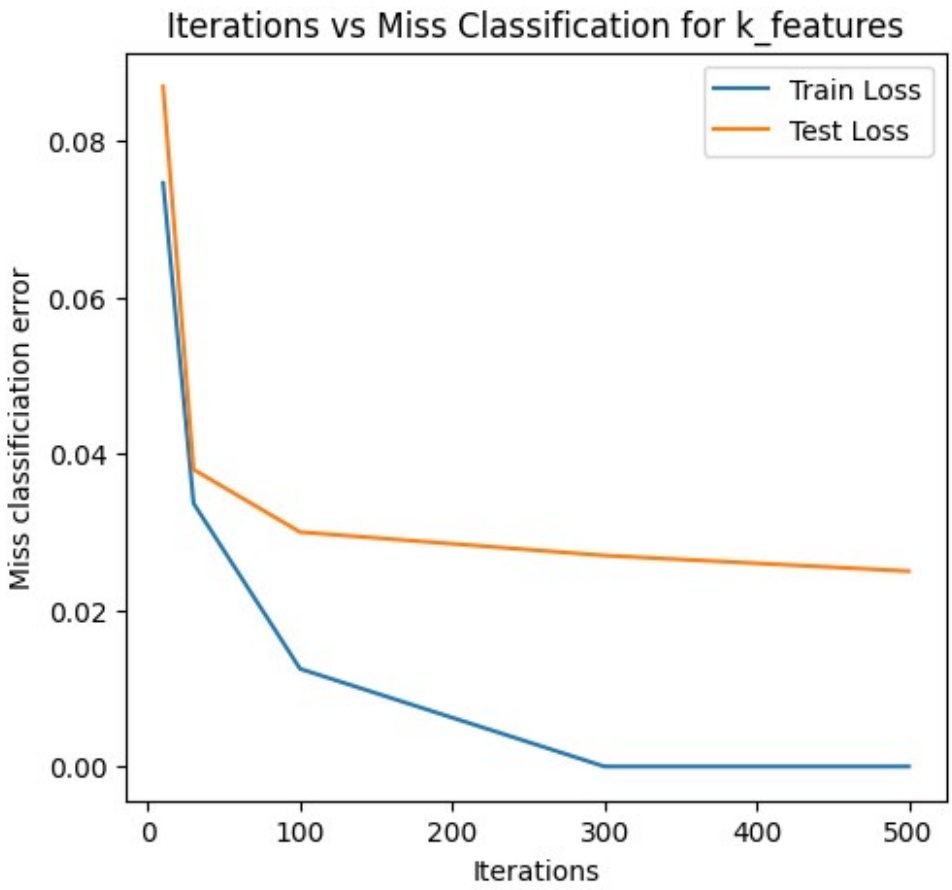
```
Xones = np.ones(X.shape[0])
Xtest_ones = np.ones(Xtest.shape[0])
X = np.insert(X, 0, Xones, axis=1)
Xtest = np.insert(Xtest, 0, Xtest_ones, axis=1)

M = X.shape[1]

k_features = [10,30,100,300,500]

train_misclass_errors = []
valid_misclass_errors = []
```

```
train_loss_300 = []
fpr_train_list = []
tpr_train_list = []
roc_auc_train_list = []
fpr_valid_list = []
tpr_valid_list = []
roc_auc_valid_list = []

Logitboost(X, y, Xtest, yTest, M, k_features, iterations)
```

Plot the stuff

```
plt.figure(figsize=(12,5))
plt.subplot(1, 2, 1)
plt.plot(range(1,301), train_loss_300, label="Train Loss")
plt.xlabel('Iterations')
plt.ylabel('Training Loss')
plt.title('Iterations vs Training Loss for Feature k=300')
plt.legend()

plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(k_features, train_misclass_errors, label="Train Loss")
plt.plot(k_features, valid_misclass_errors, label="Test Loss")
plt.xlabel('Iterations')
plt.ylabel('Miss classificiation error')
plt.title('Iterations vs Miss Classification for k_features')
plt.legend()

plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 2)
plt.plot(fpr_train_list[-1], tpr_train_list[-1], color='blue', lw=2,
label=f'Training Set (AUC = {roc_auc_train_list[-1]:.2f})')
plt.plot(fpr_valid_list[-1], tpr_valid_list[-1], color='darkorange',
lw=2, label=f'Validation Set (AUC = {roc_auc_valid_list[-1]:.2f})')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()



plt.show()
```


Iterations vs Training Loss for Feature k=300
Train Loss
Training Loss
2500
2000
1500
1000
500
0
0
50
100
150
200
250
300
Iterations


Iterations vs Miss Classification for k_features
Train Loss
Test Loss
Miss classificiation error
0.08
0.06
0.04
0.02
0.00
0
100
200
300
400
500
Iterations

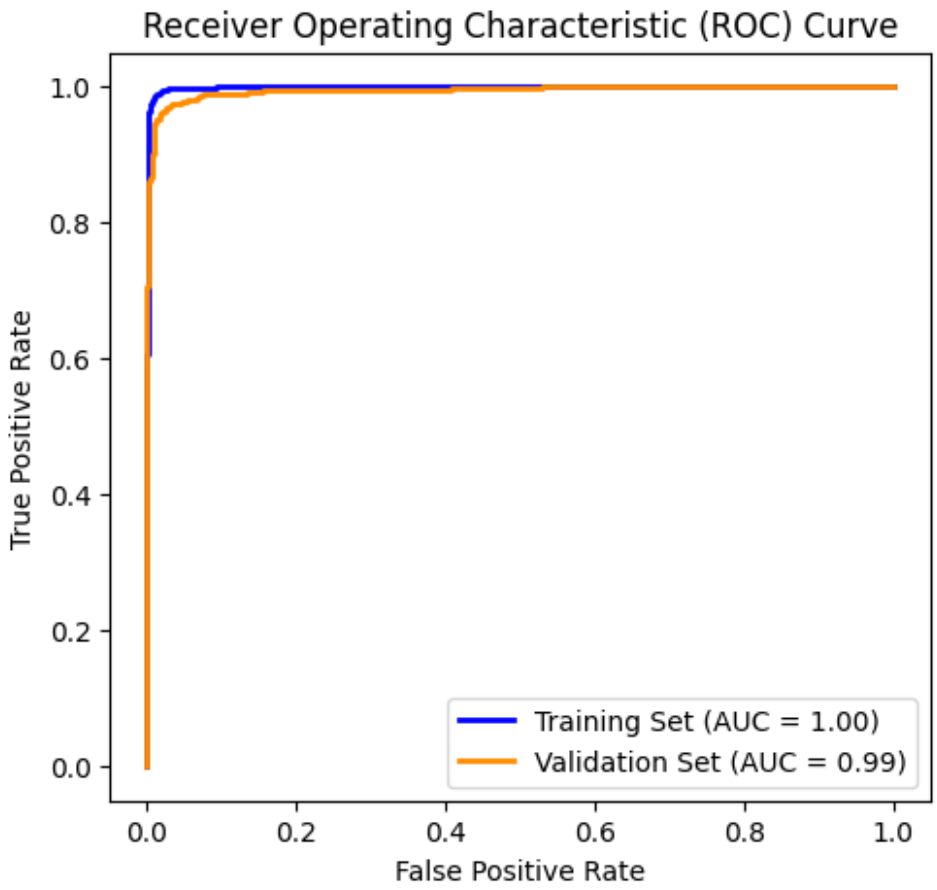


Table with the Misclass errors, features on test and training sets

```
results = pd.DataFrame({
    'K-features:': k_features,
    'Train MisClass Error:': train_misclass_errors,
    'Test MisClass Error:': valid_misclass_errors
})

print(results)
```

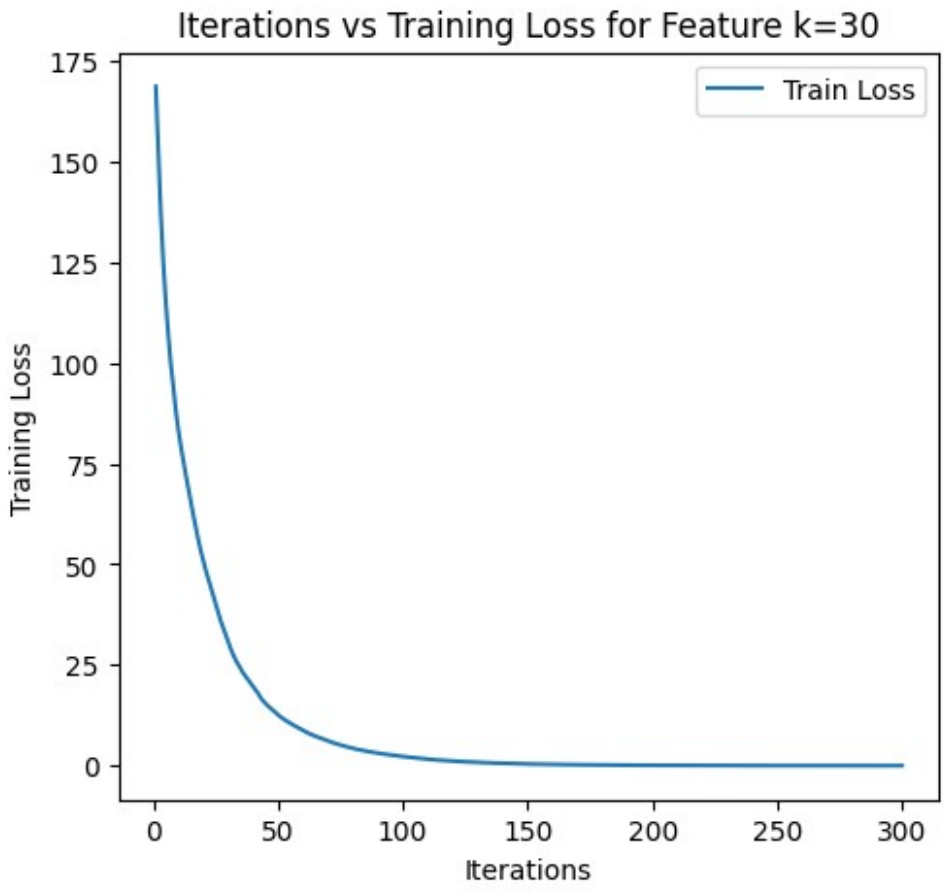
```
   K-features:  Train MisClass Error:  Test MisClass Error:
0           10                0.074667                 0.087
1           30                0.033667                 0.038
2          100                0.012500                 0.030
3          300                0.000000                 0.027
4          500                0.000000                 0.025
```

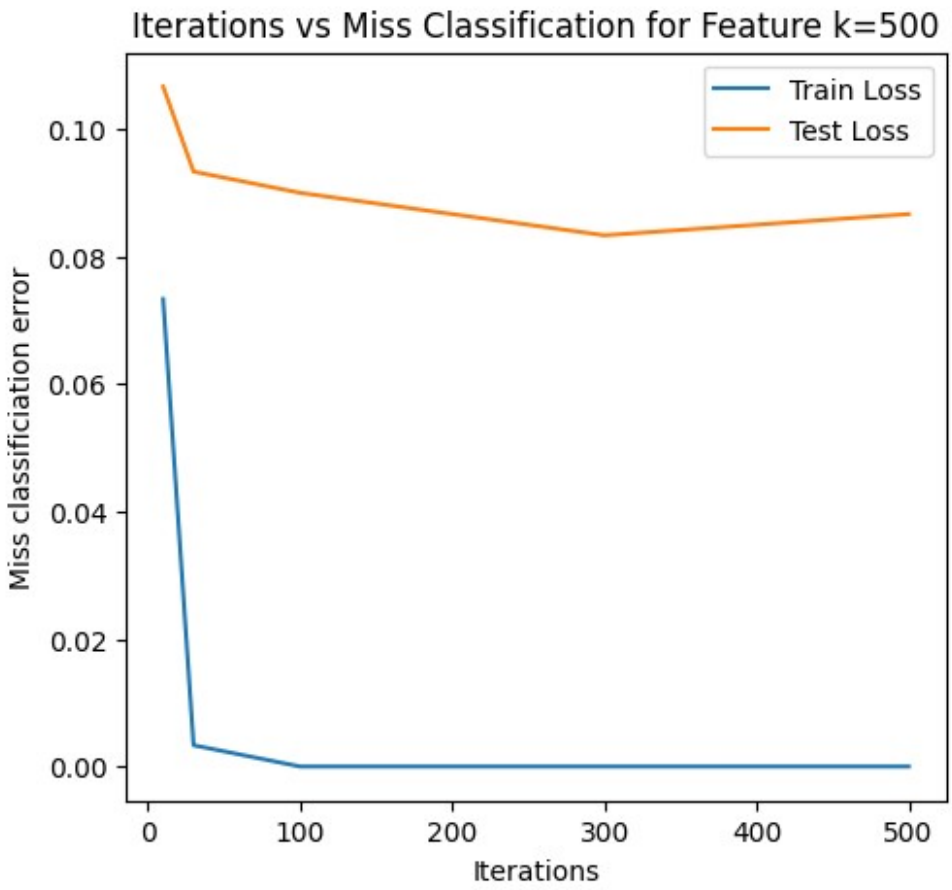
1b) DEXTER DATASET

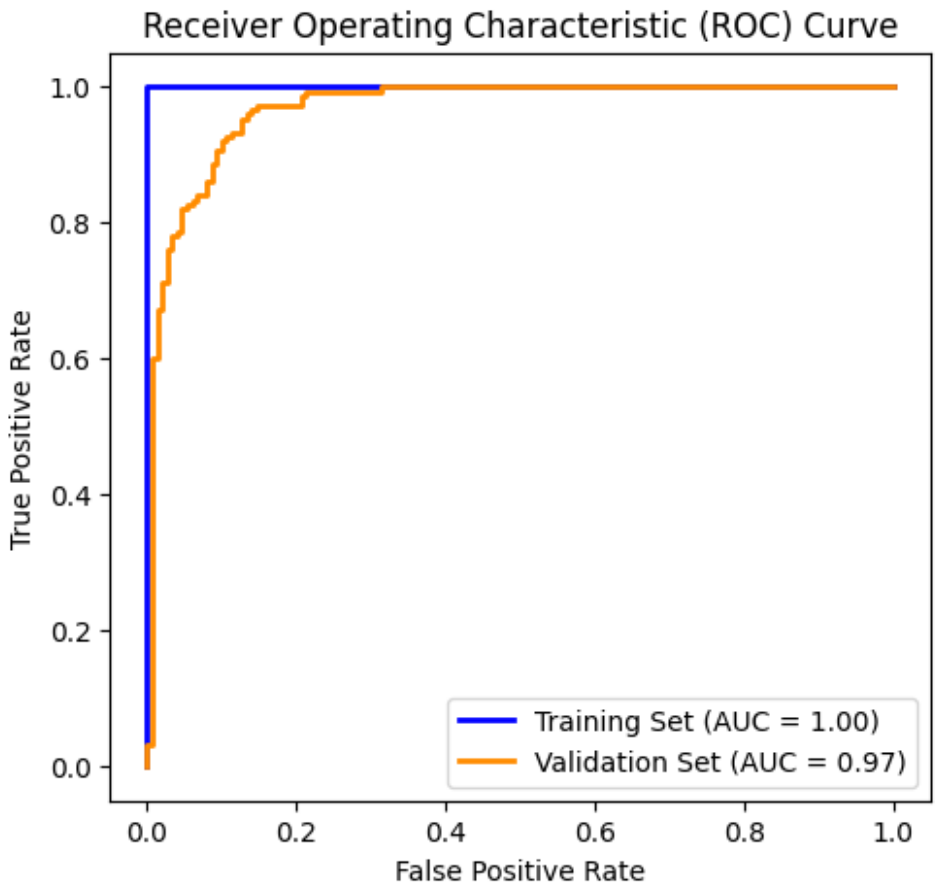
```
X = np.empty(1)
y = np.empty(1)
```

```
Xtest = np.empty(1)
yTest = np.empty(1)
probChar = 'b'

X, y, Xtest, yTest = load_dataset(X, y, Xtest, yTest, probChar)
Xones = np.ones(X.shape[0])
Xtest_ones = np.ones(Xtest.shape[0])
X = np.insert(X, 0, Xones, axis=1)
Xtest = np.insert(Xtest, 0, Xtest_ones, axis=1)

M = X.shape[1]

k_features = [10,30,100,300,500]

train_misclass_errors = []
valid_misclass_errors = []
train_loss_300 = []
fpr_train_list = []
tpr_train_list = []
roc_auc_train_list = []
fpr_valid_list = []
tpr_valid_list = []
roc_auc_valid_list = []

Logitboost(X, y, Xtest, yTest, M, k_features, iterations)

plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(range(1,301), train_loss_300, label="Train Loss")
plt.xlabel('Iterations')
plt.ylabel('Training Loss')
plt.title('Iterations vs Training Loss for Feature k=30')
plt.legend()

plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(k_features, train_misclass_errors, label="Train Loss")
plt.plot(k_features, valid_misclass_errors, label="Test Loss")
plt.xlabel('Iterations')
plt.ylabel('Miss classificiation error')
plt.title('Iterations vs Miss Classification for Feature k=500')
plt.legend()

plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 2)
plt.plot(fpr_train_list[-1], tpr_train_list[-1], color='blue', lw=2,
label=f'Training Set (AUC = {roc_auc_train_list[-1]:.2f})')
plt.plot(fpr_valid_list[-1], tpr_valid_list[-1], color='darkorange',
lw=2, label=f'Validation Set (AUC = {roc_auc_valid_list[-1]:.2f})')
plt.xlabel('False Positive Rate')
```

```
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()


plt.show()
```




Iterations vs Miss Classification for Feature k=500
Train Loss
Test Loss
Miss classificiation error
0.10
0.08
0.06
0.04
0.02
0.00
0
100
200
300
400
500
Iterations




```
results = pd.DataFrame({
    'K-features:': k_features,
    'Train MisClass Error:': train_misclass_errors,
    'Test MisClass Error:': valid_misclass_errors
})

print(results)
```

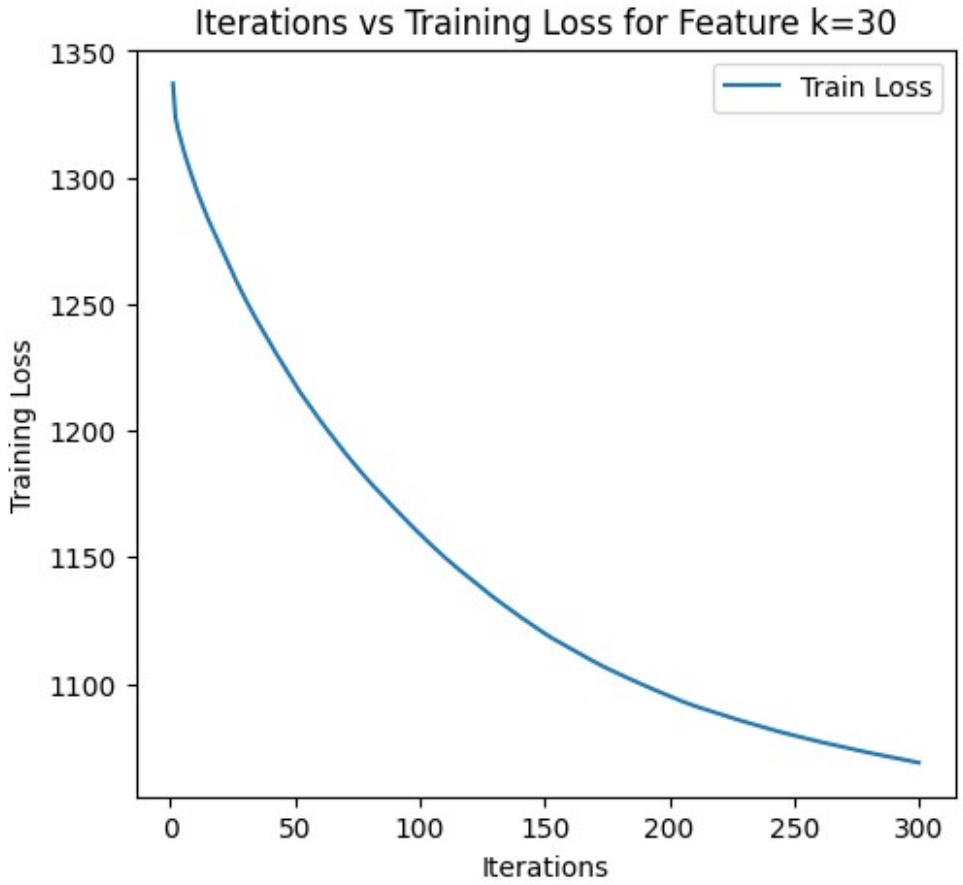
	K-features:	Train MisClass Error:	Test MisClass Error:
0	10	0.073333	0.106667
1	30	0.003333	0.093333
2	100	0.000000	0.090000
3	300	0.000000	0.083333
4	500	0.000000	0.086667

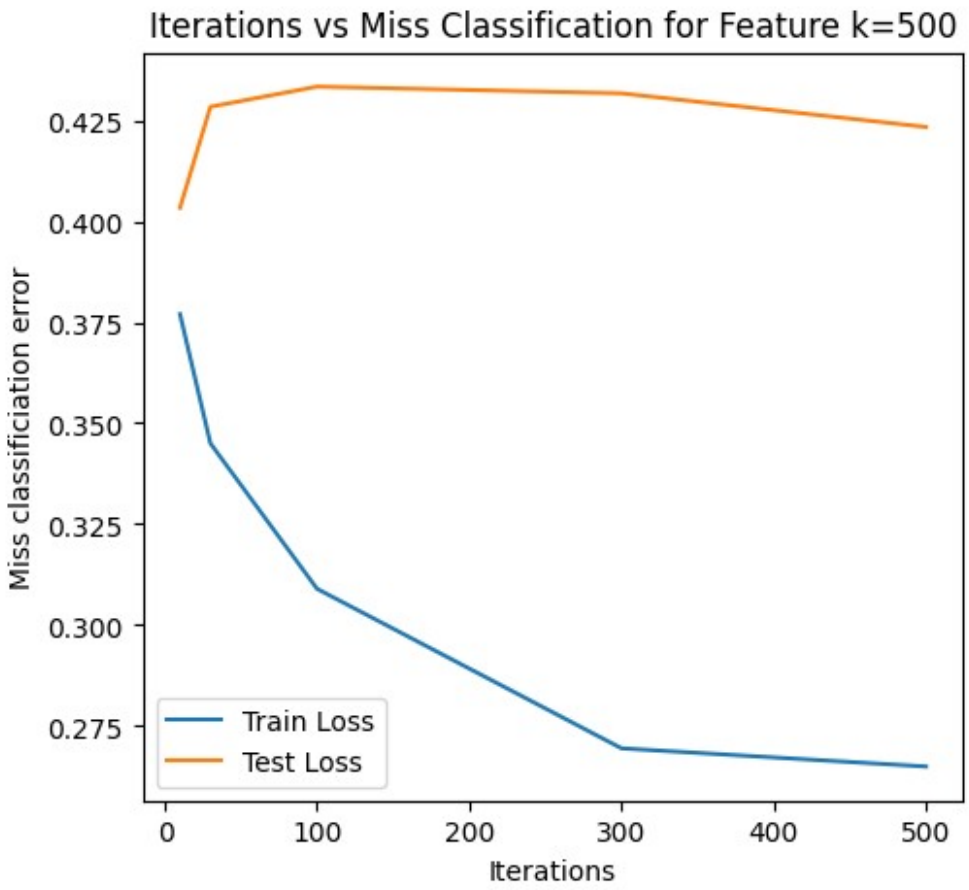
1c) MADELON DATASET

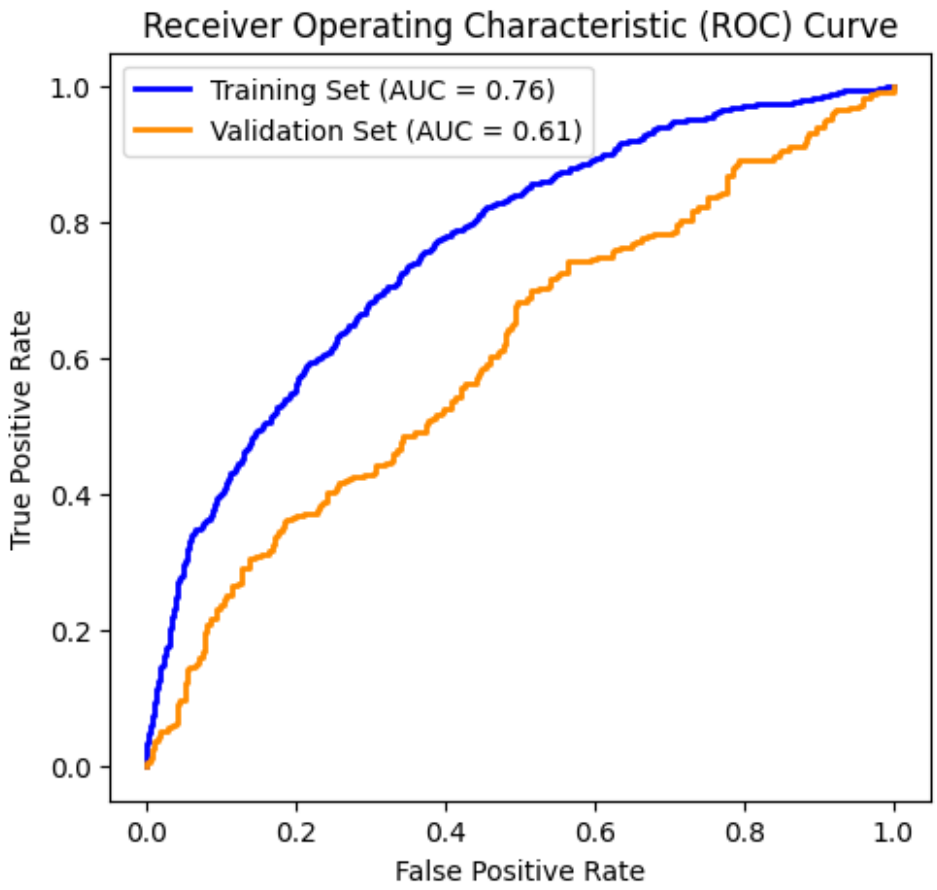
```
X = np.empty(1)
y = np.empty(1)
Xtest = np.empty(1)
```

```
yTest = np.empty(1)
probChar = 'c'

X, y, Xtest, yTest = load_dataset(X, y, Xtest, yTest, probChar)
Xones = np.ones(X.shape[0])
Xtest_ones = np.ones(Xtest.shape[0])
X = np.insert(X, 0, Xones, axis=1)
Xtest = np.insert(Xtest, 0, Xtest_ones, axis=1)

M = X.shape[1]

k_features = [10,30,100,300,500]

train_misclass_errors = []
valid_misclass_errors = []
train_loss_300 = []
fpr_train_list = []
tpr_train_list = []
roc_auc_train_list = []
fpr_valid_list = []
tpr_valid_list = []
roc_auc_valid_list = []

Logitboost(X, y, Xtest, yTest, M, k_features, iterations)

plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(range(1,301), train_loss_300, label="Train Loss")
plt.xlabel('Iterations')
plt.ylabel('Training Loss')
plt.title('Iterations vs Training Loss for Feature k=30')
plt.legend()

plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(k_features, train_misclass_errors, label="Train Loss")
plt.plot(k_features, valid_misclass_errors, label="Test Loss")
plt.xlabel('Iterations')
plt.ylabel('Miss classificiation error')
plt.title('Iterations vs Miss Classification for Feature k=500')
plt.legend()

plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 2)
plt.plot(fpr_train_list[-1], tpr_train_list[-1], color='blue', lw=2,
label=f'Training Set (AUC = {roc_auc_train_list[-1]:.2f})')
plt.plot(fpr_valid_list[-1], tpr_valid_list[-1], color='darkorange',
lw=2, label=f'Validation Set (AUC = {roc_auc_valid_list[-1]:.2f})')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```

```
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()


plt.show()
```




Iterations vs Miss Classification for Feature k=500
Miss classificiation error
0.425
0.400
0.375
0.350
0.325
0.300
0.275
Train Loss
Test Loss
0
100
200
300
400
500
Iterations




```
results = pd.DataFrame({
    'K-features:': k_features,
    'Train MisClass Error:': train_misclass_errors,
    'Test MisClass Error:': valid_misclass_errors
})

print(results)
```

	K-features:	Train MisClass Error:	Test MisClass Error:
0	10	0.3770	0.403333
1	30	0.3450	0.428333
2	100	0.3090	0.433333
3	300	0.2695	0.431667
4	500	0.2650	0.423333