In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from scipy.special import expit
from sklearn.metrics import accuracy_score, roc_curve
```

In [ ]:

```
X = pd.read_csv('D:\School\Applied ML FSU\Applied-ML-FSU\Data\Gisette\gisette_train.data', sep = ' ', header= None)
y = pd.read_csv('D:\School\Applied ML FSU\Applied-ML-FSU\Data\Gisette\gisette_train.labels', header = None)
Xt = pd.read_csv('D:\School\Applied ML FSU\Applied-ML-FSU\Data\Gisette\gisette_valid.data', sep = ' ', header = None)
yt = pd.read_csv('D:\School\Applied ML FSU\Applied-ML-FSU\Data\Gisette\gisette_valid.labels', header= None)
```

In [ ]:

```
X = X.drop(5000, axis = 1)
```

In [ ]:

```
Xt = Xt.drop(5000, axis = 1)
```

In [ ]:

```
scaler = StandardScaler()
Xs = scaler.fit_transform(X)
Xts = scaler.transform(Xt)
```

In [ ]:

```
Xs = np.hstack([np.ones((Xs.shape[0], 1)), Xs])
Xts = np.hstack([np.ones((Xts.shape[0], 1)), Xts])
```

In [ ]:

```
y = np.where(y == -1, 0, 1)
yt = np.where(yt == -1, 0, 1)
```

In [ ]:

```
def threshold(x, ld):
    return np.where(np.abs(x) < ld, 0, x)
```

In [ ]:

```
lrate = 1/Xs.shape[0]
iter = 100

w = np.zeros(Xs.shape[1])
for _ in range(iter):
    y = y.squeeze()
    Xw = np.matmul(Xs,w)
    grad = np.matmul(Xs.T, y - expit(Xw))
    w = threshold(w + lrate*grad, 0.0385)
```

In [ ]:

```
np.count_nonzero(w)
```

Out[ ]:

```
503
```

In [ ]:

```
def fit(X, y, ld = 1, iter = 100):
    lrate = 1/X.shape[0]
    w = np.zeros(X.shape[1])
    for _ in range(iter):
        y = y.squeeze()
        Xw = np.matmul(X, w)
        grad = np.matmul(X.T, y - expit(Xw))
        w = threshold(w + lrate*grad, ld)
    return w
```

In [ ]:

```
def fit_w_iter(X,y, ld = 1, iter = 100):
    w_iter = []
    lrate = 1/X.shape[0]
    w = np.zeros(X.shape[1])
    for _ in range(iter):
        y = y.squeeze()
        Xw = np.matmul(X, w)
        grad = np.matmul(X.T, y - expit(Xw))
        w = threshold(w + lrate*grad, ld)
        w_iter.append(w)
    return w_iter
```

In [ ]:

```
def n_feat(w):
    return np.count_nonzero(w)
```

In [ ]:

```
w_500 = fit(Xs, y, ld = 0.0385)
n_feat(w_500)
```

Out[ ]:

```
503
```

In [ ]:

```
w_300 = fit(Xs, y, ld = 0.053)
n_feat(w_300)
```

Out[ ]:

```
299
```

In [ ]:

```python
w_100 = fit(Xs, y, ld = 0.088)
n_feat(w_100)
```

Out[ ]:

```
98
```

In [ ]:

```python
w_30 = fit(Xs, y, ld = 0.133)
n_feat(w_30)
```

Out[ ]:

```
30
```

In [ ]:

```python
w_10 = fit(Xs, y, ld = 0.19)
n_feat(w_10)
```

Out[ ]:

```
10
```

In [ ]:

```python
Xw_10 = np.matmul(Xs, w_10)
pred_y_10 = np.where(expit(Xw_10) > 0.5, 1, 0)
accuracy_score(pred_y_10, y)
```

Out[ ]:

```
0.8786666666666667
```

In [ ]:

```python
def score(X, y, w):
    Xw = np.matmul(X,w)
    p = np.where(expit(Xw) > 0.5, 1, 0)
    return accuracy_score(p, y)
```

In [ ]:

```python
score(Xs, y, w_10)
```

Out[ ]:

```
0.8786666666666667
```

In [ ]:

```python
def misclass(X,y,w):
    Xw = np.matmul(X,w)
    pred = np.where(expit(Xw) > 0.5, 1, 0)
    return (1 - accuracy_score(pred, y))
```

In [ ]:

```python
misclass(Xs, y, w_10)
```

Out[ ]:

```
0.1213333333333333
```

In [ ]:

```python
w_100_iter = fit_w_iter(Xs, y, ld = 0.088)
```

In [ ]:

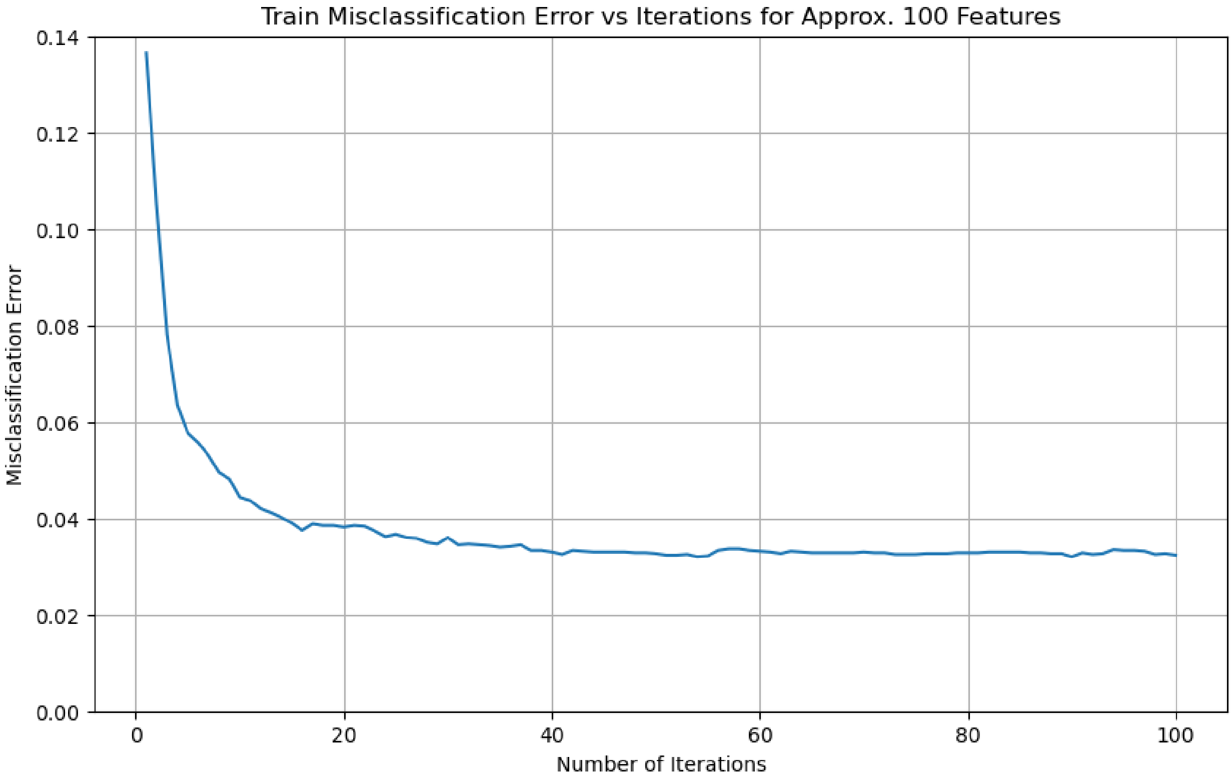
```python
def misclass_w_iter(X, y, w_iter):
    mis = []
    for w in w_iter:
        mis.append(1 - score(X,y,w))
    return mis
```

In [ ]:

```python
mis_100_w = misclass_w_iter(Xs, y, w_100_iter)
```

In [ ]:

```python
plt.figure(figsize=(10,6))
plt.plot(range(1,101), mis_100_w)
plt.grid()
plt.title('Train Misclassification Error vs Iterations for Approx. 100 Features')
plt.xlabel('Number of Iterations')
plt.ylabel('Misclassification Error')
plt.ylim(0, 0.14)
plt.show()
```

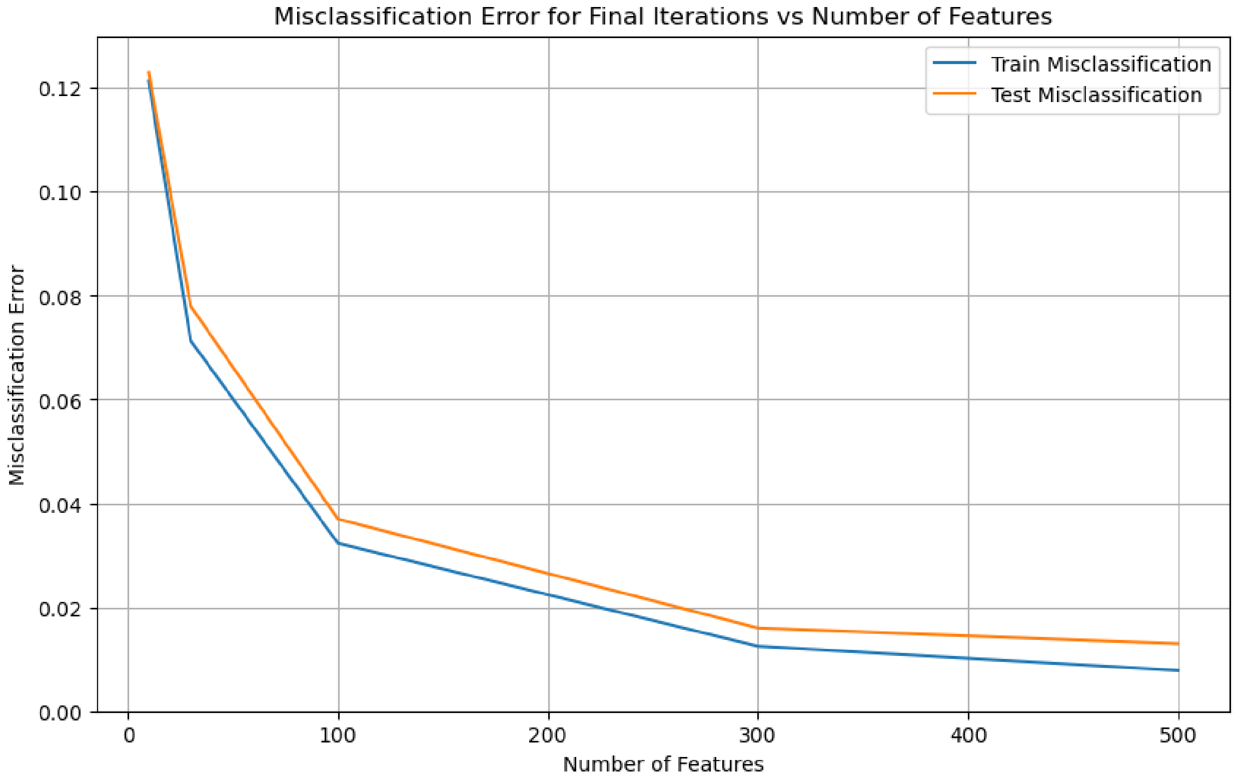


In [ ]:

```python
fin_w = [w_10, w_30, w_100, w_300, w_500]
misclass_fin_train = []
misclass_fin_test = []
for w in fin_w:
    misclass_fin_train.append(misclass(Xs, y, w))
    misclass_fin_test.append(misclass(Xts, yt, w))
```

In [ ]:

```python
plt.figure(figsize=(10,6))
plt.plot([10,30,100,300,500], misclass_fin_train, label = 'Train Misclassification')
plt.plot([10,30,100,300,500], misclass_fin_test, label = 'Test Misclassification')
plt.title('Misclassification Error for Final Iterations vs Number of Features')
plt.xlabel('Number of Features')
plt.ylabel('Misclassification Error')
plt.grid()
plt.ylim(0,0.13)
plt.legend()
plt.show()
```



In [ ]:

```
misclass_fin_test
```

Out[ ]:

```
[0.123,
 0.07799999999999996,
 0.03700000000000003,
 0.016000000000000014,
 0.013000000000000012]
```

In [ ]:

```
misclass_fin_train
```

Out[ ]:

```
[0.12133333333333333,
 0.07133333333333336,
 0.032333333333333325,
 0.012499999999999956,
 0.007833333333333359]
```

In [ ]:

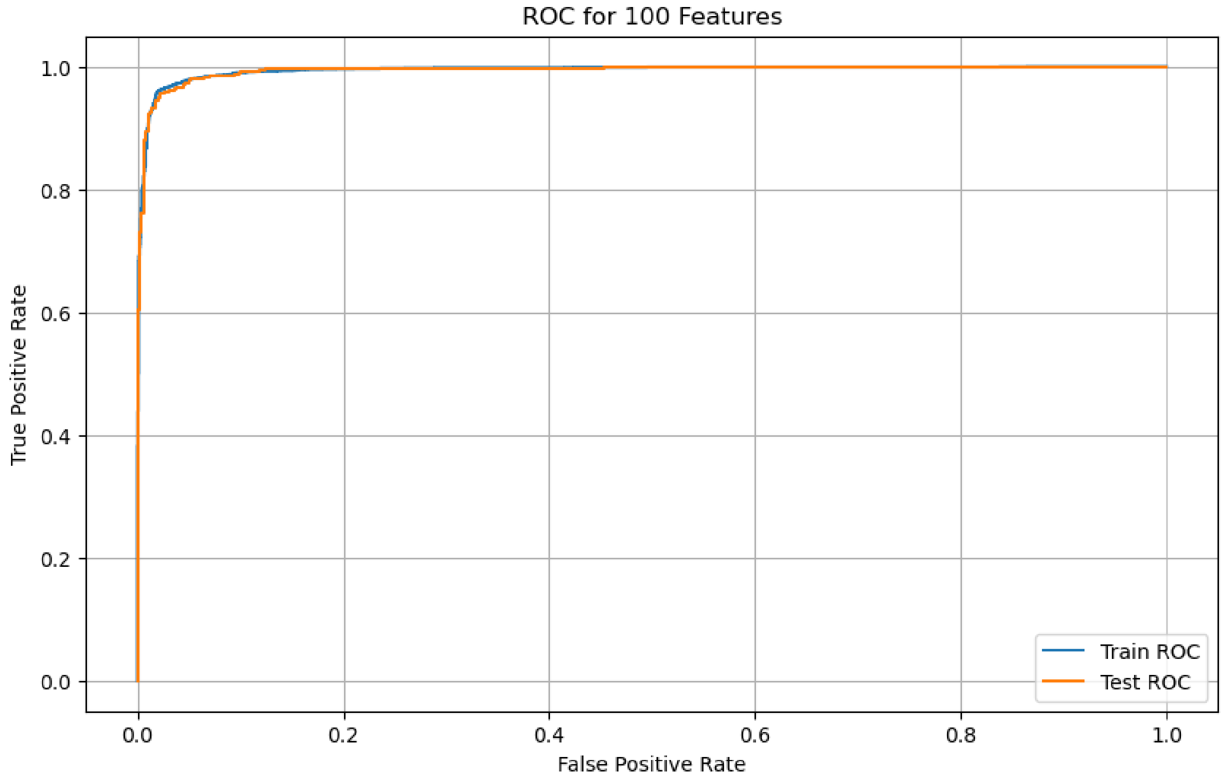
```
prob_y_100 = expit(np.matmul(Xs, w_100))
prob_yt_100 = expit(np.matmul(Xts, w_100))
```

In [ ]:

```
fpr_train, tpr_train, _ = roc_curve(y, prob_y_100)
fpr_test, tpr_test, _ = roc_curve(yt, prob_yt_100)
```

In [ ]:

```
plt.figure(figsize=(10,6))
plt.plot(fpr_train, tpr_train, label= 'Train ROC')
plt.plot(fpr_test, tpr_test, label = 'Test ROC')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC for 100 Features')
plt.grid()
plt.legend()
plt.show()
```



Madelon Dataset

In [ ]:

```
X = pd.read_csv('D:\School\Applied ML FSU\Applied-ML-FSU\Data\MADELON\madelon_train.data', sep = ' ', header= None)
y = pd.read_csv('D:\School\Applied ML FSU\Applied-ML-FSU\Data\MADELON\madelon_train.labels', header = None)
Xt = pd.read_csv('D:\School\Applied ML FSU\Applied-ML-FSU\Data\MADELON\madelon_valid.data', sep = ' ', header = None)
yt = pd.read_csv('D:\School\Applied ML FSU\Applied-ML-FSU\Data\MADELON\madelon_valid.labels', header= None)
```

In [ ]:

```
X = X.drop(500, axis = 1)
Xt = Xt.drop(500, axis= 1)
```

In [ ]:

```
scaler = StandardScaler()
Xs = scaler.fit_transform(X)
Xts = scaler.transform(Xt)
```

In [ ]:

```
Xs = np.hstack([np.ones((Xs.shape[0], 1)), Xs])
Xts = np.hstack([np.ones((Xts.shape[0], 1)), Xts])
```

In [ ]:

```
y = np.where(y == -1, 0, 1)
yt = np.where(yt == -1, 0, 1)
```

In [ ]:

```
w_10 = fit(Xs, y, ld = 0.0298)
n_feat(w_10)
```

Out[ ]:

```
8
```

In [ ]:

```
w_30 = fit(Xs, y, ld = 0.0242)
n_feat(w_30)
```

Out[ ]:

```
31
```

In [ ]:

```
w_100 = fit(Xs, y, ld = 0.017)
n_feat(w_100)
```

Out[ ]:

```
101
```

In [ ]:

```
w_300 = fit(Xs, y, ld = 0.00735)
n_feat(w_300)
```

Out[ ]:

```
302
```

In [ ]:

```
w_500 = fit(Xs, y, ld = 0.00002)
n_feat(w_500)
```

Out[ ]:

```
501
```

In [ ]:

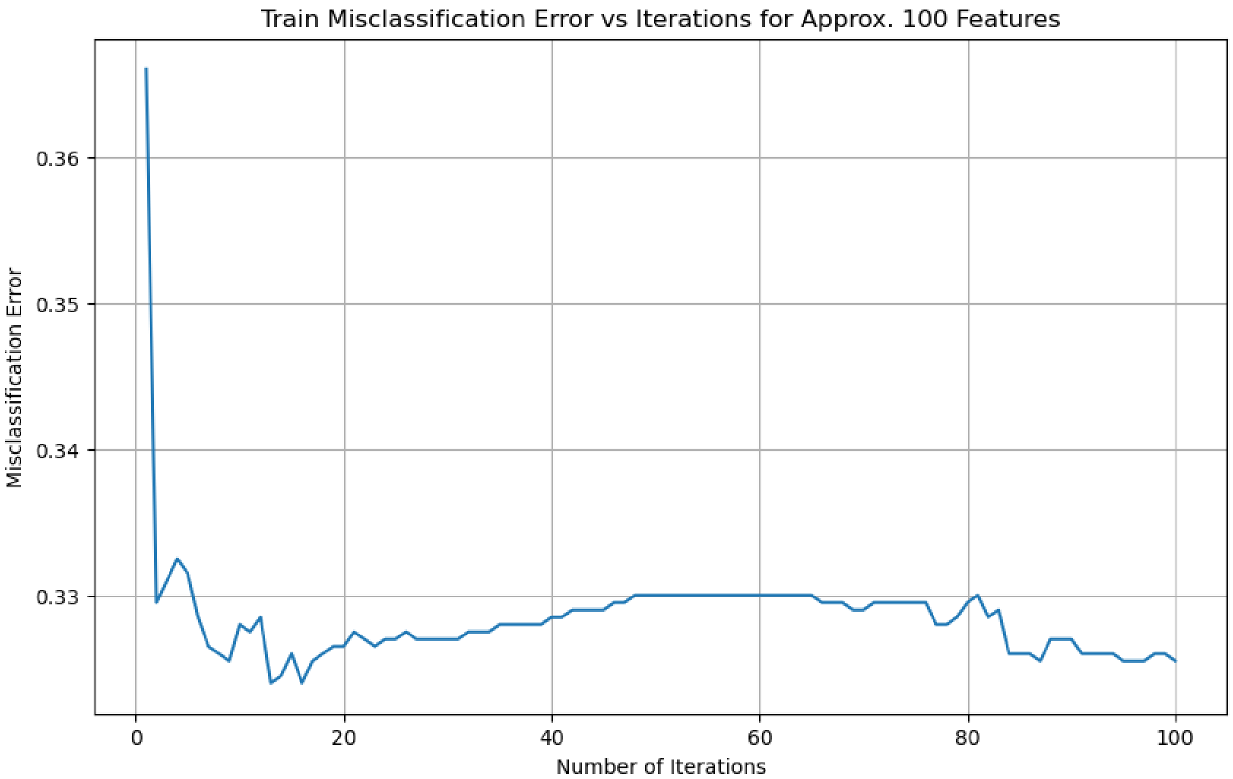
```
w_100_iter = fit_w_iter(Xs, y, ld = 0.017)
```

In [ ]:

```
mis_100_w = misclass_w_iter(Xs, y, w_100_iter)
```

In [ ]:

```
plt.figure(figsize=(10,6))
plt.plot(range(1,101), mis_100_w)
plt.grid()
plt.title('Train Misclassification Error vs Iterations for Approx. 100 Features')
plt.xlabel('Number of Iterations')
plt.ylabel('Misclassification Error')
plt.show()
```

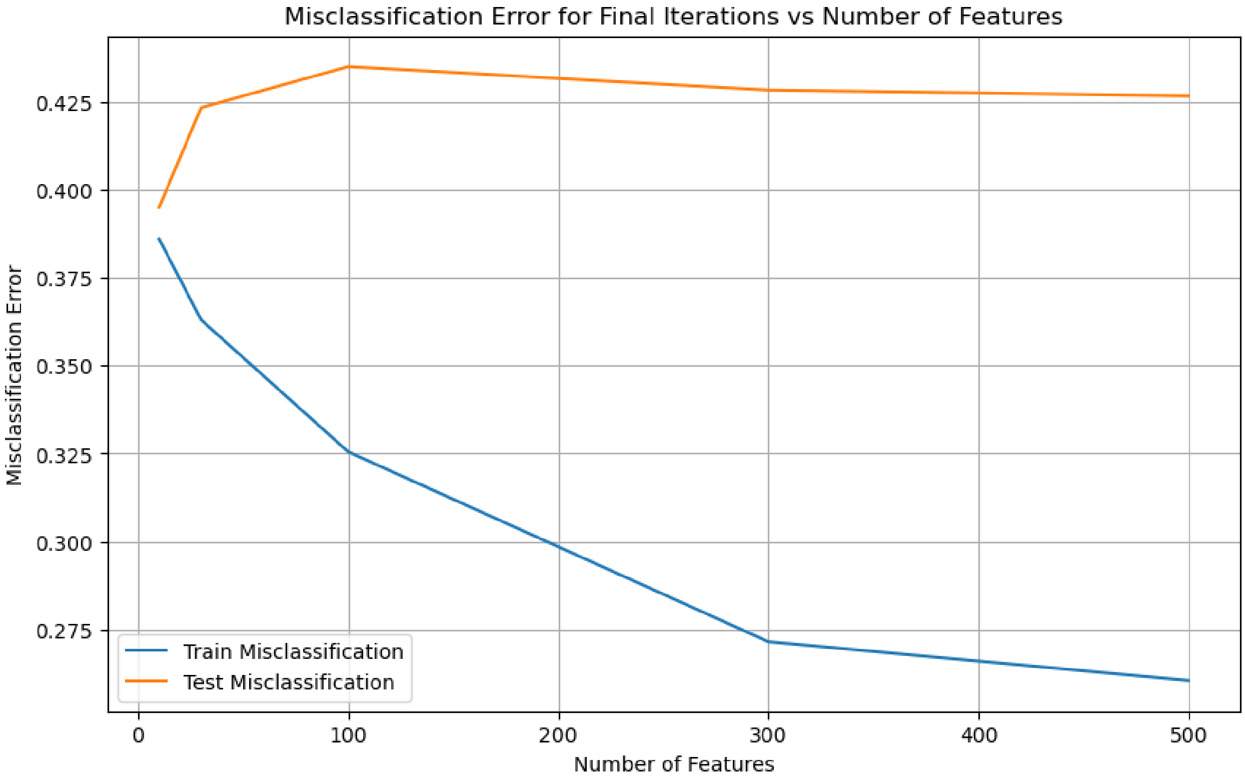


In [ ]:

```
fin_w = [w_10, w_30, w_100, w_300, w_500]
misclass_fin_train = []
misclass_fin_test = []
for w in fin_w:
    misclass_fin_train.append(misclass(Xs, y, w))
    misclass_fin_test.append(misclass(Xts, yt, w))
```

In [ ]:

```
plt.figure(figsize=(10,6))
plt.plot([10,30,100,300,500], misclass_fin_train, label = 'Train Misclassification')
plt.plot([10,30,100,300,500], misclass_fin_test, label = 'Test Misclassification')
plt.title('Misclassification Error for Final Iterations vs Number of Features')
plt.xlabel('Number of Features')
plt.ylabel('Misclassification Error')
plt.grid()
plt.legend()
plt.show()
```



In [ ]:

```
misclass_fin_test
```

Out[ ]:

```
[0.395,
 0.42333333333333334,
 0.43500000000000005,
 0.42833333333333334,
 0.42666666666666664]
```

In [ ]:

```
misclass_fin_train
```

Out[ ]:

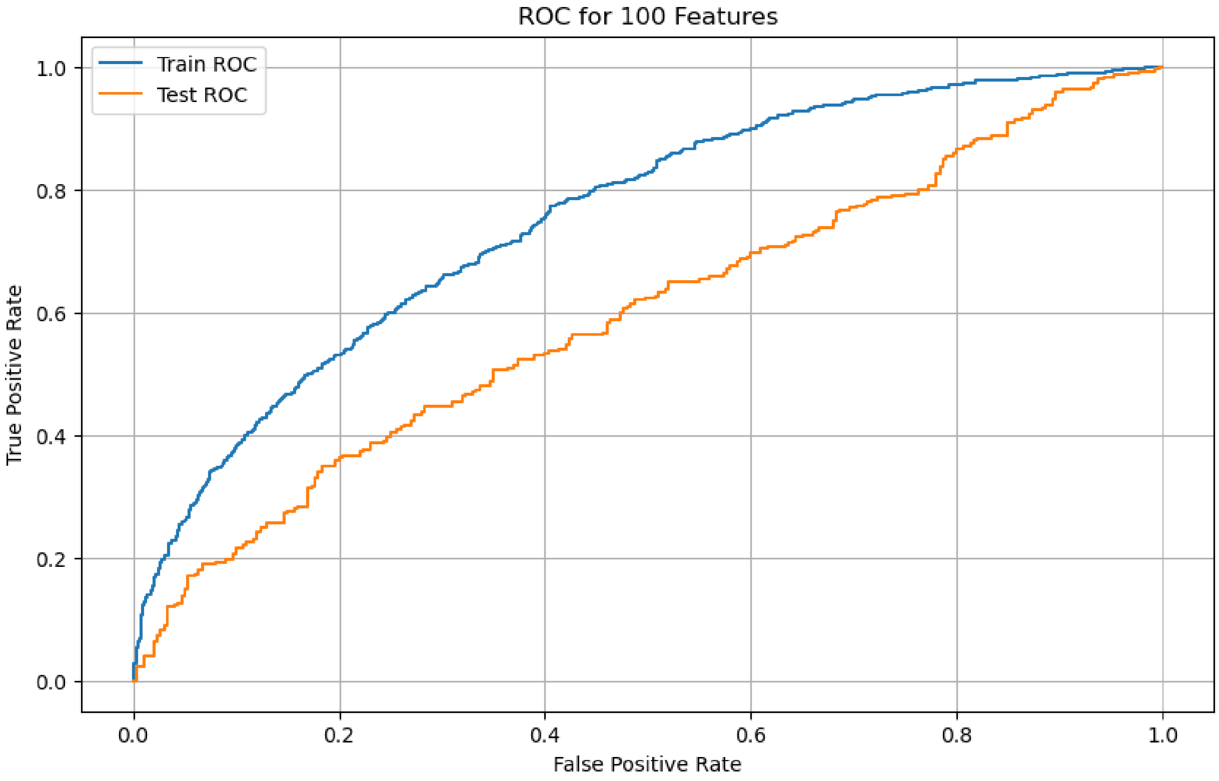
```
[0.386, 0.363, 0.3255, 0.27149999999999996, 0.26049999999999995]
```

In [ ]:

```
prob_y_100 = expit(np.matmul(Xs, w_100))
prob_yt_100 = expit(np.matmul(Xts, w_100))
fpr_train, tpr_train, _ = roc_curve(y, prob_y_100)
fpr_test, tpr_test, _ = roc_curve(yt, prob_yt_100)
```

In [ ]:

```
plt.figure(figsize=(10,6))
plt.plot(fpr_train, tpr_train, label= 'Train ROC')
plt.plot(fpr_test, tpr_test, label = 'Test ROC')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC for 100 Features')
plt.grid()
plt.legend()
plt.show()
```



Dexter Dataset

```
In [ ]: X = pd.read_csv('D:\School\Applied ML FSU\Applied-ML-FSU\Data\Dexter\dexter_train.csv', header= None)
        y = pd.read_csv('D:\School\Applied ML FSU\Applied-ML-FSU\Data\Dexter\dexter_train.labels', header = None)
        Xt = pd.read_csv('D:\School\Applied ML FSU\Applied-ML-FSU\Data\Dexter\dexter_valid.csv', header= None)
        yt = pd.read_csv('D:\School\Applied ML FSU\Applied-ML-FSU\Data\Dexter\dexter_valid.labels', header = None)
```

```
In [ ]: scaler = StandardScaler()
        Xs = scaler.fit_transform(X)
        Xts = scaler.transform(Xt)
```

```
In [ ]: Xs = np.hstack([np.ones((Xs.shape[0], 1)), Xs])
        Xts = np.hstack([np.ones((Xts.shape[0], 1)), Xts])
```

```
In [ ]: y = np.where(y == -1, 0, 1)
        yt = np.where(yt == -1, 0, 1)
```

```
In [ ]: w_10 = fit(Xs, y, ld = 0.14)
        n_feat(w_10)

Out[ ]: 11
```

```
In [ ]: w_30 = fit(Xs, y, ld = 0.099)
        n_feat(w_30)

Out[ ]: 29
```

```
In [ ]: w_100 = fit(Xs, y, ld = 0.071)
        n_feat(w_100)

Out[ ]: 101
```

```
In [ ]: w_300 = fit(Xs, y, ld = 0.0523)
        n_feat(w_300)

Out[ ]: 299
```

```
In [ ]: w_500 = fit(Xs, y, ld = 0.0466)
        n_feat(w_500)

Out[ ]: 500
```

In [ ]:

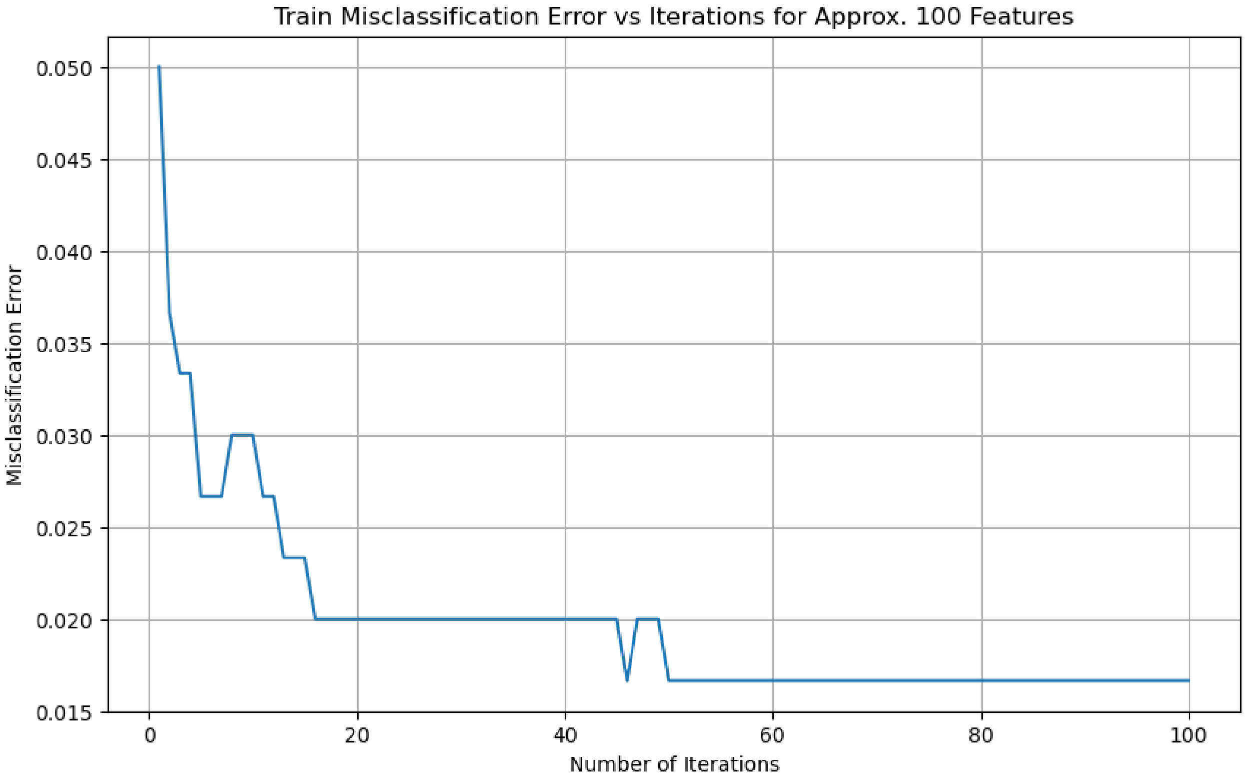
```
w_100_iter = fit_w_iter(Xs, y, ld = 0.071)
```

In [ ]:

```
mis_100_w = misclass_w_iter(Xs, y, w_100_iter)
```

In [ ]:

```
plt.figure(figsize=(10,6))
plt.plot(range(1,101), mis_100_w)
plt.grid()
plt.title('Train Misclassification Error vs Iterations for Approx. 100 Features')
plt.xlabel('Number of Iterations')
plt.ylabel('Misclassification Error')
plt.show()
```

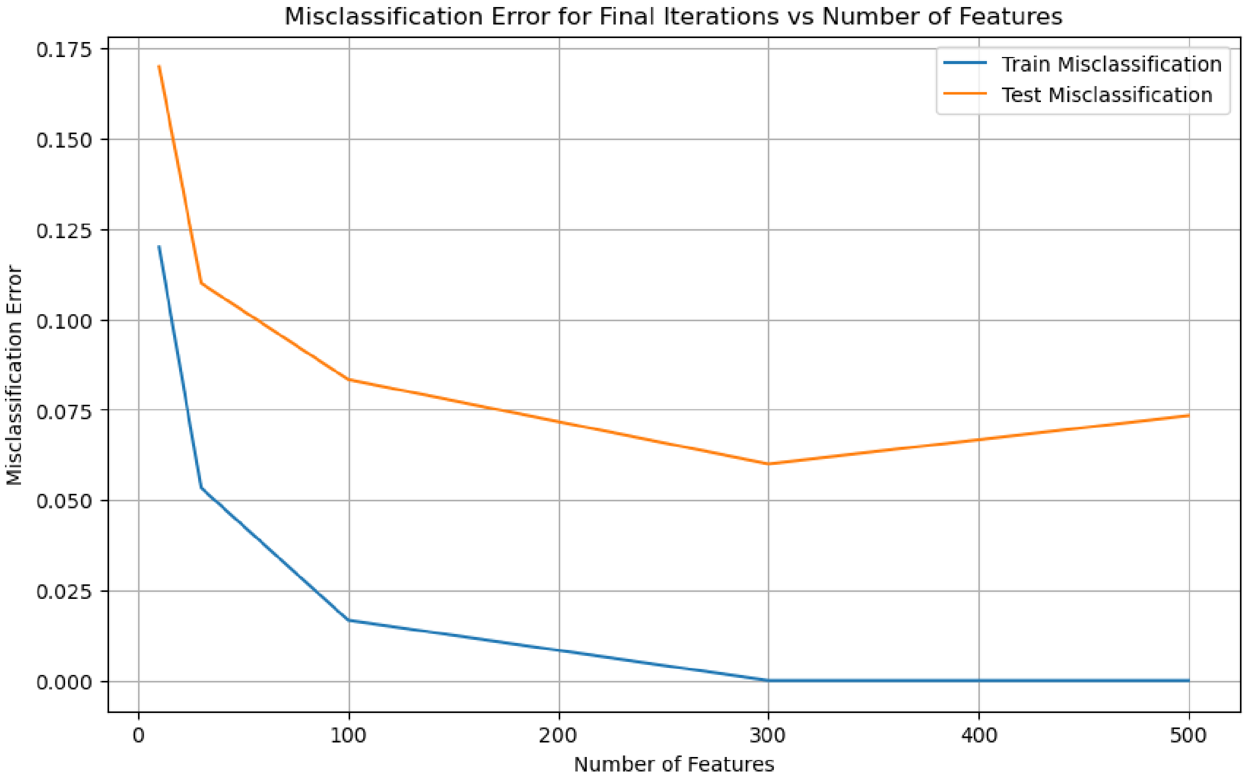


In [ ]:

```
fin_w = [w_10, w_30, w_100, w_300, w_500]
misclass_fin_train = []
misclass_fin_test = []
for w in fin_w:
    misclass_fin_train.append(misclass(Xs, y, w))
    misclass_fin_test.append(misclass(Xts, yt, w))
```

In [ ]:

```
plt.figure(figsize=(10,6))
plt.plot([10,30,100,300,500], misclass_fin_train, label = 'Train Misclassification')
plt.plot([10,30,100,300,500], misclass_fin_test, label = 'Test Misclassification')
plt.title('Misclassification Error for Final Iterations vs Number of Features')
plt.xlabel('Number of Features')
plt.ylabel('Misclassification Error')
plt.grid()
plt.legend()
plt.show()
```



In [ ]:

```
misclass_fin_test
```

Out[ ]:

```
[0.17000000000000004,
 0.10999999999999999,
 0.08333333333333337,
 0.06000000000000005,
 0.07333333333333336]
```

In [ ]:

```
misclass_fin_train
```

Out[ ]:

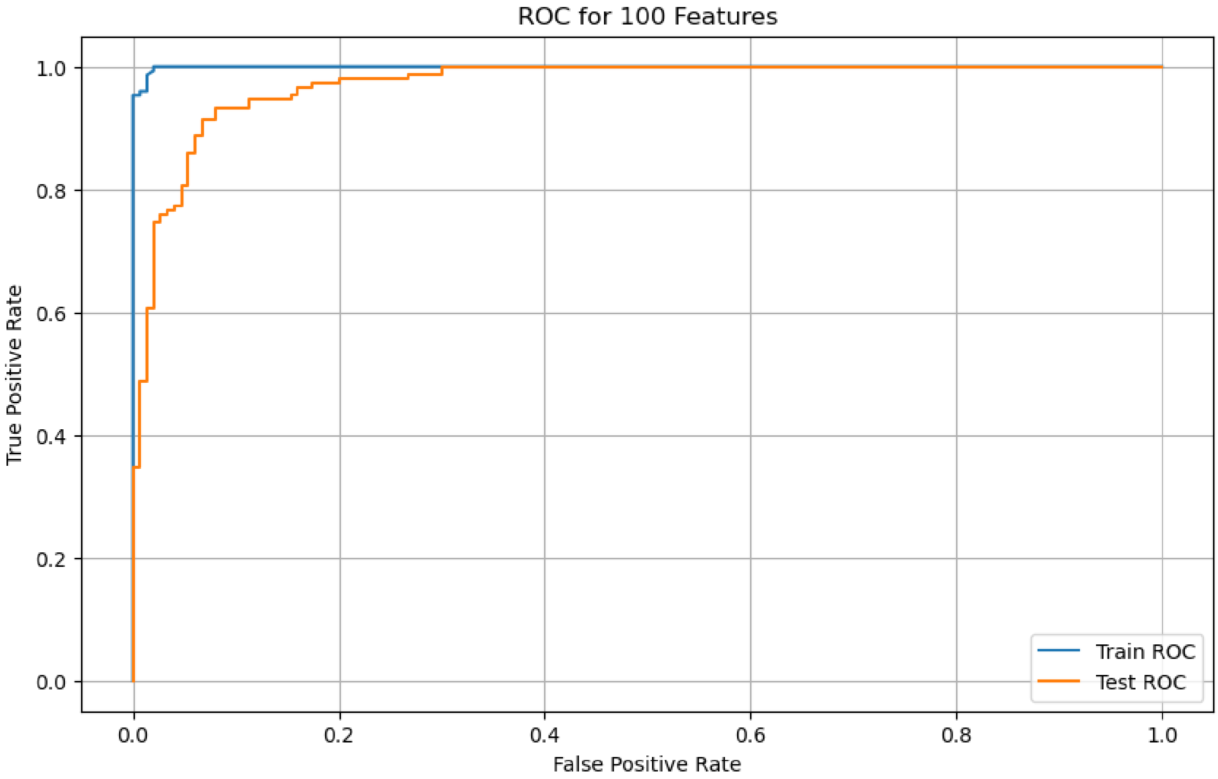
```
[0.12, 0.053333333333333344, 0.01666666666666672, 0.0, 0.0]
```

In [ ]:

```
prob_y_100 = expit(np.matmul(Xs, w_100))
prob_yt_100 = expit(np.matmul(Xts, w_100))
fpr_train, tpr_train, _ = roc_curve(y, prob_y_100)
fpr_test, tpr_test, _ = roc_curve(yt, prob_yt_100)
```

In [ ]:

```
plt.figure(figsize=(10,6))
plt.plot(fpr_train, tpr_train, label= 'Train ROC')
plt.plot(fpr_test, tpr_test, label = 'Test ROC')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC for 100 Features')
plt.grid()
plt.legend()
plt.show()
```


ROC for 100 Features
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
Train ROC
Test ROC