MACHINE LEARNING HOMEWORK 4

		Q1	Total
Grade	Max	5	5 points
	Expected	5	5

STUDENT

NAME SURNAME : ALPEREN KANTARCI

STUDENT NUMBER : 504191504

DEADLINE : 19/12/2019

COURSE NAME : MACHINE LEARNING

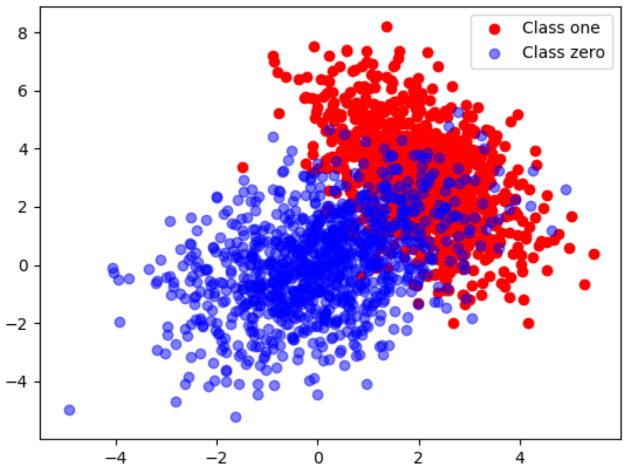
INSTRUCTOR : YUSUF YASLAN

QUESTION 1:

First i have created sigmoid function but because of the high values overflow may happen and therefore i applied some normalization trick.

```
def exp_normalize(x):
    b = x.max()
    y = np.exp(x - b)
    return y / y.sum()

def sigmoid(x):
    z = 1/(1 + exp_normalize(-x))
    return z
```

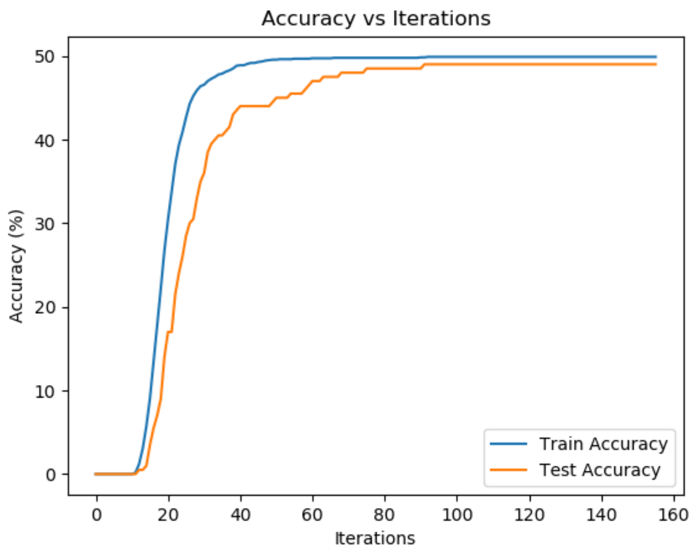


I plotted the data points to see the distribution of the dataset. It looks like linearly seperable dataset but even with the linear boundary there would be many errors.

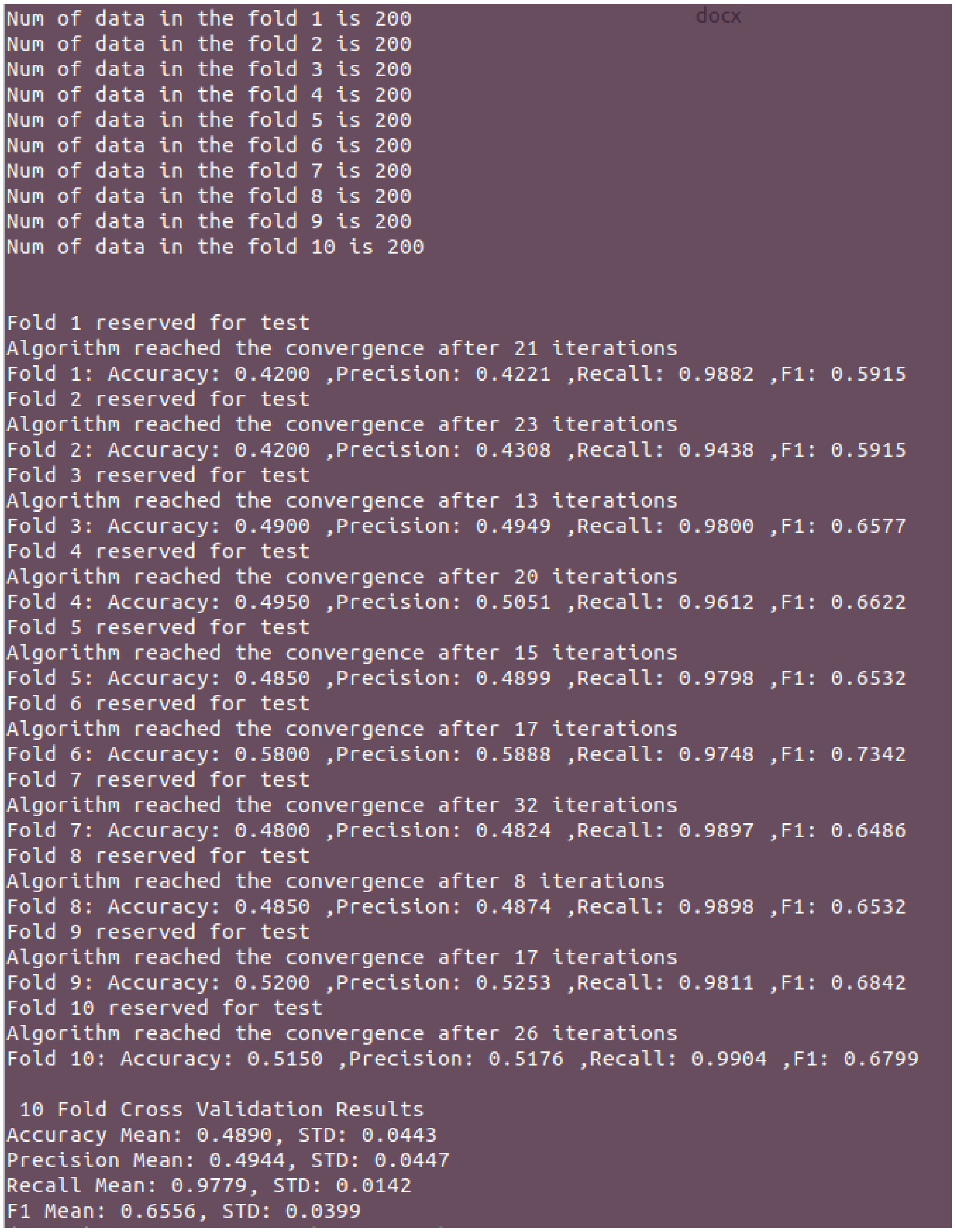
Then I have applied logistic discriminant algorithm from the Alpaydın’s book but given figure implements iterative algorithm. I have vectorized the given algorithm for faster calculation.

```
def logistic_discriminant(data,calculate_err_each_iter=False,test_data=None):
    train_errors = []
    test_errors = []
    num_points = data.shape[0]
    num_features = len(data[0,:-1])
    w = np.full((1,num_features),random.uniform(-0.01,0.01))
    convergence_point = 1e-18
    old_delta_w = np.full((1,num_features),0)
    count = 1
    while True:
        delta_w = np.full((1,num_features),0)
        o = np.dot(w,data[:,:-1].T)
        y = sigmoid(o)
        if calculate_err_each_iter:
            train_errors.append(tuple(evaluate_classification(data,y)))
            o_test = np.dot(w,test_data[:,:-1].T)
            y_test = sigmoid(o_test)
            test_errors.append(tuple(evaluate_classification(test_data,y_test)))
        # Calculate Delta_w
        delta_w = delta_w + np.dot((np.expand_dims(data[:,-1], axis=0)-y),data[:,:-1])
        w = w - 0.2*delta_w
        dif = np.abs(np.sum(old_delta_w - delta_w))
        if np.sum(dif) < convergence_point:
            print("Algorithm reached the convergence after {} iterations".format(count))
            break
        old_delta_w = delta_w.copy()
        count += 1
    return w, train_errors, test_errors
```

Algorithm stops when weights do not change more than 1e^-18 which is a fairly good indication that algorithm reached to stable point.



Below you can see the results of the algorithm on 10 fold cross validation. And also at the above you will see the iteration vs accuracy plot of the last fold.



```
Num of data in the fold 1 is 200
Num of data in the fold 2 is 200
Num of data in the fold 3 is 200
Num of data in the fold 4 is 200
Num of data in the fold 5 is 200
Num of data in the fold 6 is 200
Num of data in the fold 7 is 200
Num of data in the fold 8 is 200
Num of data in the fold 9 is 200
Num of data in the fold 10 is 200


Fold 1 reserved for test
Algorithm reached the convergence after 21 iterations
Fold 1: Accuracy: 0.4200 ,Precision: 0.4221 ,Recall: 0.9882 ,F1: 0.5915
Fold 2 reserved for test
Algorithm reached the convergence after 23 iterations
Fold 2: Accuracy: 0.4200 ,Precision: 0.4308 ,Recall: 0.9438 ,F1: 0.5915
Fold 3 reserved for test
Algorithm reached the convergence after 13 iterations
Fold 3: Accuracy: 0.4900 ,Precision: 0.4949 ,Recall: 0.9800 ,F1: 0.6577
Fold 4 reserved for test
Algorithm reached the convergence after 20 iterations
Fold 4: Accuracy: 0.4950 ,Precision: 0.5051 ,Recall: 0.9612 ,F1: 0.6622
Fold 5 reserved for test
Algorithm reached the convergence after 15 iterations
Fold 5: Accuracy: 0.4850 ,Precision: 0.4899 ,Recall: 0.9798 ,F1: 0.6532
Fold 6 reserved for test
Algorithm reached the convergence after 17 iterations
Fold 6: Accuracy: 0.5800 ,Precision: 0.5888 ,Recall: 0.9748 ,F1: 0.7342
Fold 7 reserved for test
Algorithm reached the convergence after 32 iterations
Fold 7: Accuracy: 0.4800 ,Precision: 0.4824 ,Recall: 0.9897 ,F1: 0.6486
Fold 8 reserved for test
Algorithm reached the convergence after 8 iterations
Fold 8: Accuracy: 0.4850 ,Precision: 0.4874 ,Recall: 0.9898 ,F1: 0.6532
Fold 9 reserved for test
Algorithm reached the convergence after 17 iterations
Fold 9: Accuracy: 0.5200 ,Precision: 0.5253 ,Recall: 0.9811 ,F1: 0.6842
Fold 10 reserved for test
Algorithm reached the convergence after 26 iterations
Fold 10: Accuracy: 0.5150 ,Precision: 0.5176 ,Recall: 0.9904 ,F1: 0.6799

 10 Fold Cross Validation Results
Accuracy Mean: 0.4890, STD: 0.0443
Precision Mean: 0.4944, STD: 0.0447
Recall Mean: 0.9779, STD: 0.0142
F1 Mean: 0.6556, STD: 0.0399
```