```python
import numpy as np
import matplotlib.pyplot as plt

def fix_target(target,n_classes):
    n_target = len(target)
    new_target = np.zeros((n_target,n_classes))
    for i in range(n_target):
        vector_target = np.zeros((1,n_classes))[0]
        vector_target[target[i]]+=1
        new_target[i]=vector_target
    new_target = np.reshape(new_target,(len(target),n_classes))
    return new_target

def sigmoid(z):
    return 1/(1+ np.exp(-z))

def calculate_MSE(gk,tk):
    return 0.5*np.matmul((gk-tk).T,(gk-tk))

def calculate_grad_W_MSE(g, t, x):
    return np.matmul(((g-t)*g*(1-g)).T,x.T)


def _removeFeature(data, features, featureToBeRemoved):
    n_features = len(features)
    newFeature = np.array([]) ##hard to preallocate string as you need to know the size, bad for efficiancy but whatever
    n_data = len(data)
    newData = np.array([[0.]*(n_features-1) for j in range(n_data)])

    j = 0 #ugly hack to get correct index for newFeature
    for i in range(n_features):
        if i%n_features != featureToBeRemoved:
            newFeature = np.append(newFeature, features[i])
            j+=1

    for i in range(n_data):
        k = 0
        for j in range(n_features):
            if j%n_features != featureToBeRemoved:
                newData[i][k] = data[i][j]
                k+=1

    return newData, newFeature

def removeListOfFeatures(data, feature, featureRemoveList):
    newData = data
    newFeature = feature
    for i in range(len(featureRemoveList)):
        newIndex = np.where(newFeature == feature[i])[0][0]
        newData, newFeature = _removeFeature(newData, newFeature, newIndex)

    return newData, newFeature

def hist(data, features, classes, file = 0):
    histData = data.T
```

```
    fig = plt.figure()
    plt.suptitle('Histograms of the different features and classes', fontweight='bold')

    featuresLeftToPlot = len(features)
    for f in range(len(features)):
        if featuresLeftToPlot>=2:
            xdir = 2
        else:
            xdir = 1
        plt.subplot(int(np.ceil(len(features)/2)), xdir, f+1)
        for c in range(3):
            plt.hist(histData[f][c*50:(c+1)*50], bins=30, alpha=0.5, label=classes[c])

        plt.title(features[f])
        plt.legend(loc='upper right')

    # Adding a plot in the figure which will encapsulate all the subplots with axis showing only
    fig.add_subplot(1, 1, 1, frame_on=False)

    # Hiding the axis ticks and tick labels of the bigger plot
    plt.tick_params(labelcolor="none", bottom=False, left=False)

    #Make common x- and y-labels
    plt.xlabel('Length (cm)', fontweight='bold')
    plt.ylabel('Occurrences', fontweight='bold')

    if file != 0:
        plt.savefig(file)

    plt.show()
    return
```