In [66]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plot
from sklearn.utils import shuffle

def load_dataset(dataset):
    dataFrame=pd.read_csv(dataset)
    print(dataFrame.info())
    dataFrame=shuffle(dataFrame);
    dataFrame = np.asmatrix(dataFrame, dtype = 'float64')
    return dataFrame

def perceptron_algorithm(dataset, iter_count):
    output_labels = dataset[:, -1]
    input_features = dataset[:, :-1]
    dimension=input_features.shape[1]
    wt = np.zeros((1,dimension+1))
    misclassified_dt = []
    for iteration in range(iter_count):
        misclassified_count = 0
        for input_feature, label in zip(input_features, output_labels):
            input_feature = np.insert(input_feature,0,1)
            calculated_y = np.dot(wt, input_feature.transpose())
            predicted_y = 1.0 if (calculated_y > 0) else 0.0
            actual_y=label.item(0,0)
            difference = (actual_y - predicted_y)
            if(difference):
                wt += (difference * input_feature)
                misclassified_count += 1

        misclassified_dt.append(misclassified_count)
    return (wt, misclassified_dt)

def learningGraph(iter_count,misclassified_dt):
    no_of_iters = np.arange(1,iter_count+1)
    plot.plot(no_of_iters, misclassified_dt)
    plot.xlabel('No.of.epochs')
    plot.ylabel('misclassified_data')
    plot.show()

def predict(test_data, wt_vector):
    input_features = test_data[:, :-1]
    output_labels = test_data[:, -1]
    count=0
    for input_feature, label in zip(input_features, output_labels):
        input_feature = np.insert(input_feature,0,1)
        cal_y = np.dot(wt_vector, input_feature.transpose())
        pred_y = 1.0 if (cal_y > 0) else 0.0
        if(label[0]!=pred_y):
            count=count+1;
    return count

def getAccuracy(total,false_count):
    return (total-false_count)*100/total;
```

In [67]:

```
data = load_dataset("data_banknote_authentication.txt")
```

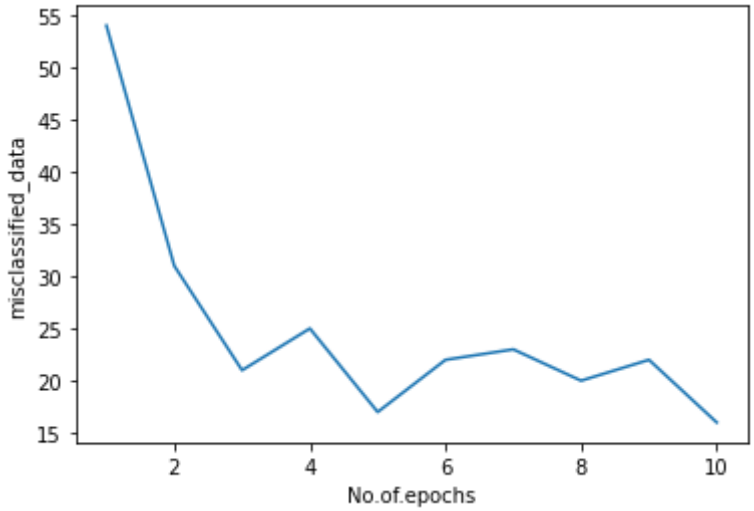
```
print(data)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1371 entries, 0 to 1370
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   3.6216    1371 non-null   float64
 1   8.6661    1371 non-null   float64
 2   -2.8073   1371 non-null   float64
 3   -0.44699  1371 non-null   float64
 4   0         1371 non-null   int64  
dtypes: float64(4), int64(1)
memory usage: 53.7 KB
None
[[ 0.51947 -3.2633   3.0895  -0.98492  0.     ]
 [ 2.2526   9.9636  -3.1749  -2.9944   0.     ]
 [-0.47465 -4.3496   1.9901   0.7517   1.     ]
 ...
 [ 3.3577  -4.3062   6.0241   0.18274  0.     ]
 [ 0.74841  7.2756   1.1504  -0.5388   0.     ]
 [-1.8411  10.8306   2.769   -3.0901   0.     ]]
```

In [68]:

```
num_iter = 10
total_records=data.shape[0];
test_count=int((total_records*30)/100);
train_count=int((total_records*60)/100);
wt_vector, misclassified_ = perceptron_algorithm(data[:train_count],10)
print("final weight vector",wt_vector)
# print(misclassified_)
learningGraph(num_iter,misclassified_)
mismatch=predict(data[test_count*-1:],wt_vector);
print("accuracy=",getAccuracy(test_count,mismatch))
```

```
final weight vector [[ 51.       -65.39035  -36.480088 -43.74535   -2.375894]]
```




```
accuracy= 99.02676399026764
```

In [ ]: