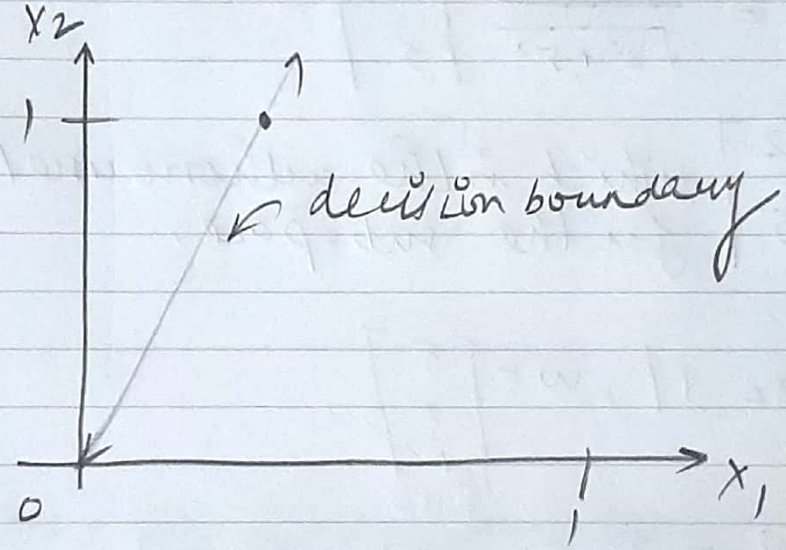
Ayan Deep Hazra, ECE 532, Activity 8
(.ipynb pdf at end)

1. a) $x^T = [x_1 \ x_2]$ & $w = \begin{bmatrix} 5 \\ -2 \end{bmatrix}$

i) $x^T w = 0 \Rightarrow 5x_1 - 2x_2 = 0$




ii) * Yes, the decision boundary represents a subspace in $\mathbb{R}^2$.

* Since, the decision boundary is a line that passes through the origin, we know that it is a subspace in $\mathbb{R}^2$.

If we take $x_1 = 2, x_2 = 5$, we find a solution to the system.

Let $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$

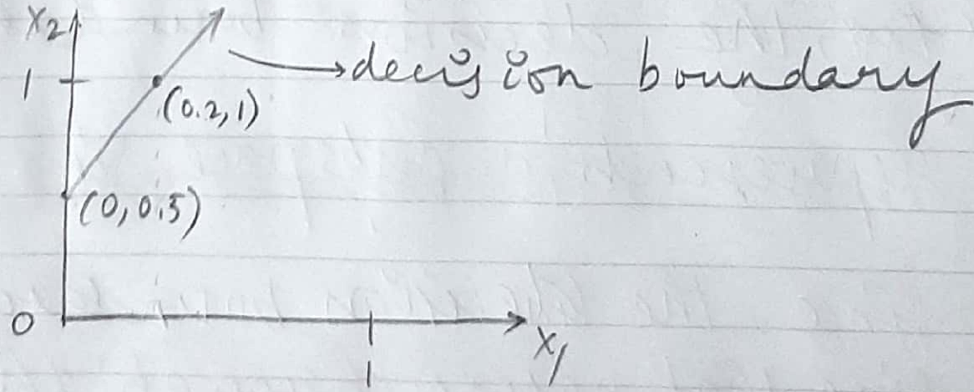
By Gram-Schmidt orthogonalization, we have

$$\vec{u}_1 = \frac{\vec{x}}{\|\vec{x}\|_2} = \frac{1}{\sqrt{2^2+5^2}}\begin{bmatrix} 2 \\ 5 \end{bmatrix}$$

$$\vec{u}_1 = \frac{1}{\sqrt{29}}\begin{bmatrix} 2 \\ 5 \end{bmatrix}$$ which is the orthonormal basis for the subspace.

b) $x^T = \begin{bmatrix} x_1 & x_2 & 1 \end{bmatrix}$, $w = \begin{bmatrix} 5 \\ -2 \\ 1 \end{bmatrix}$

i) $x^T w = 0 \Rightarrow 5x_1 - 2x_2 + 1 = 0$

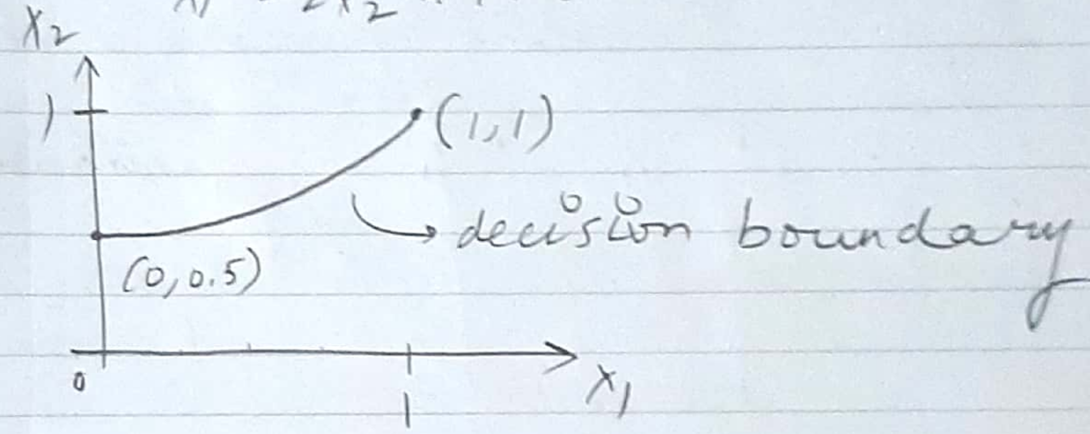



ii) The decision boundary does not represent a subspace in $\mathbb{R}^2$ as it does not pass through the origin.

c) $x^T = [x_1^2 \quad x_2 \quad 1]$, $w = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$

i) $x^T w = 0 \Rightarrow$

$$x_1^2 - 2x_2 + 1 = 0$$




ii) The decision boundary does not represent a subspace in $\mathbb{R}^2$.

This is because the boundary is not a straight line, instead it is a parabola.

Scanned with CamScanner

2.a) The actual evaluation data seems to have a more non linear curved decision boundary.

The classifier however fits a linear decision boundary.

$$\% \text{ Error} = \frac{\text{Number of errors}}{\text{Total datapoints}}$$

$$= \frac{1102}{10000}$$

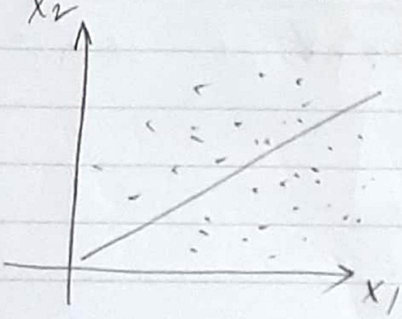
$$= 11.02\%$$

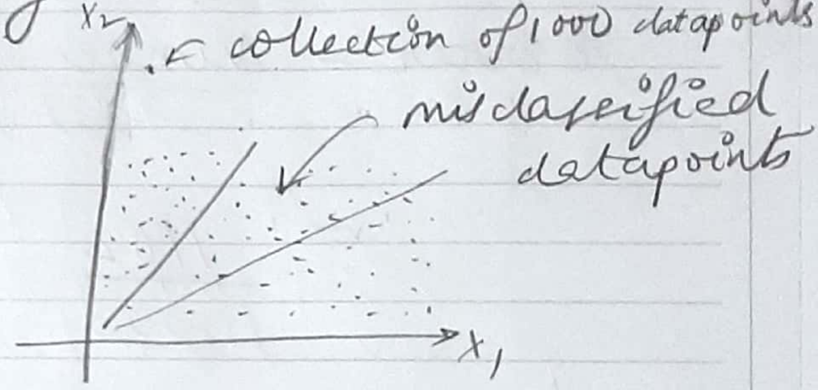
b) The curved decision boundary fits much better than the linear classifier. It captures the non-linear characteristic of the dataset much better. The error percent is now,

$$= \frac{\text{Number of errors}}{\text{Total datapoints}}$$

$$= \frac{542}{10000} = 5.42\%$$

c) Adding 1000 datapoints at $(x_1, x_2) = (0, 3)$ skews the dataset and makes the classifier predict wrong and set the decision boundary much more steep with respect to previous,




The error rate at $x_1 = 0, x_2 = 3$ is 21.34%.
& the error rate at $x_1 = 0, x_2 = 10$ is 32.77%

This happens because the farther these select datapoints are, from the actual dataset, the more they affect the remaining data and thus skewing the decision boundary.

In this case, the decision boundary becomes more steep & thus classifying more red datapoints as blue.

3. a to e) on the pdf

3. f) We know that the classifier defined in (e) is the worst as it has the most number of errors.

This is because it overfits the data and thus is bad for unseen/new data.

It overfits the data by capturing noise/disturbances in the solution for the decision boundary.

4. If we have

$$x^T = [x_1 \; x_2 \; x_3] \; \& \text{ some } w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

we see that $x^T w = 0$,

$$[x_1 \; x_2 \; x_3] \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = 0 \text{ , gives us}$$

$$x_1 w_1 + x_2 w_2 + x_3 w_3 = 0$$

the equation of a 2-D plane in the $(x_1, x_2, x_3)$ coordinate system.

5. If we define $w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}$

Given $x^T = [x_1 \ x_2 \ x_3 \ 1]$

Thus the decision boundary $x^T w = 0$ is

$$[x_1 \ x_2 \ x_3 \ 1] \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = 0 \Rightarrow x_1 w_1 + x_2 w_3 + x_3 w_3 + w_4 = 0$$

Since the plane is parallel to $x_1 - x_2$ plane we know it does not depend on $x_1$ & $x_2$ coordinates in $x^T$.

Thus, $w_1 = w_2 = 0$

Thus, $x_1 w_1 + x_2 w_2 + x_3 w_3 + w_4 = 0$,

$\Rightarrow$ (1) $w_3 = -w_4$ let $w_3 = -w_4 = k$

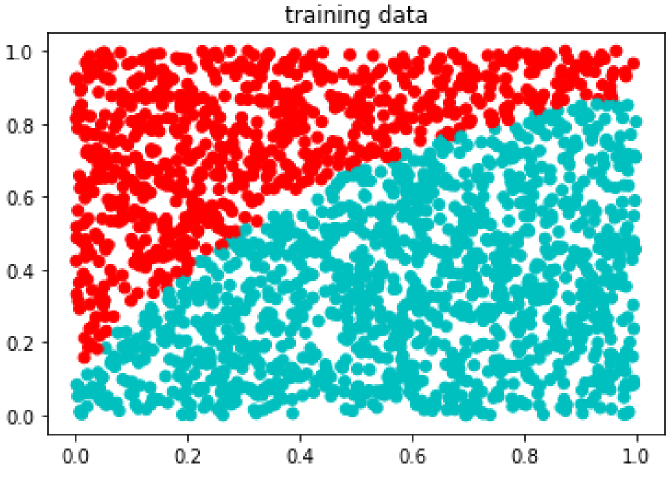
Thus, we have

$$w = \begin{bmatrix} 0 \\ 0 \\ k \\ -k \end{bmatrix} \text{ for some } k \in R$$

# 2a)

In [1]:

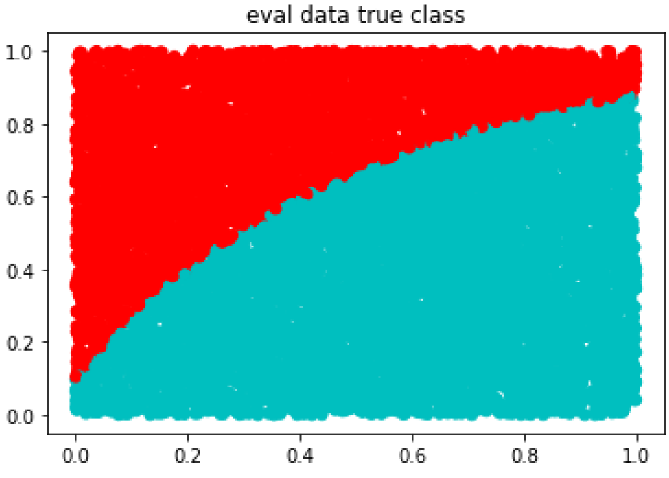
```
import numpy as np
from scipy.io import loadmat
import matplotlib.pyplot as plt

in_data = loadmat('classifier_data.mat')
#print([key for key in in_data]) # -- use this line to see the keys in the dictionary dat
a structure

x_train = in_data['x_train']
x_eval = in_data['x_eval']
y_train = in_data['y_train']
y_eval = in_data['y_eval']

n_eval = np.size(y_eval)
n_train = np.size(y_train)

plt.scatter(x_train[:,0],x_train[:,1], color=['c' if i==-1 else 'r' for i in y_train[:,0
]])
plt.title('training data')
plt.show()
print(n_eval)
```




10000

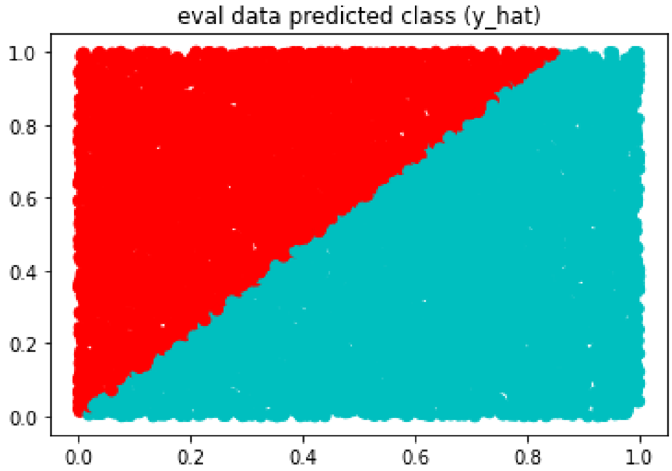
In [2]:

```
plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==-1 else 'r' for i in y_eval[:,0]])
plt.title('eval data true class')
plt.show()
```



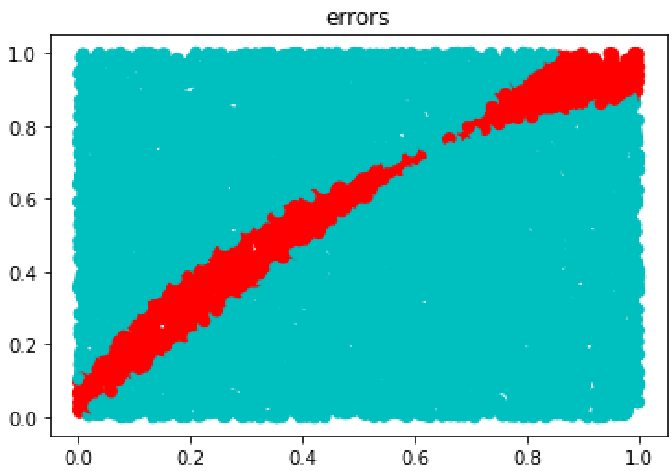
In [3]:

```
## Classifier 1

# w = (X^T X)^(-1)X^T y
w_opt = np.linalg.inv(x_train.transpose()@x_train)@x_train.transpose()@y_train
y_hat = np.sign(x_eval@w_opt)

plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==-1 else 'r' for i in y_hat[:,0]])
plt.title('eval data predicted class (y_hat)')
plt.show()
```




In [4]:

```
error_vec = [0 if i[0]==i[1] else 1 for i in np.hstack((y_hat, y_eval))]
plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==0 else 'r' for i in error_vec])
plt.title('errors')
plt.show()

print('Errors: '+ str(sum(error_vec)))
```



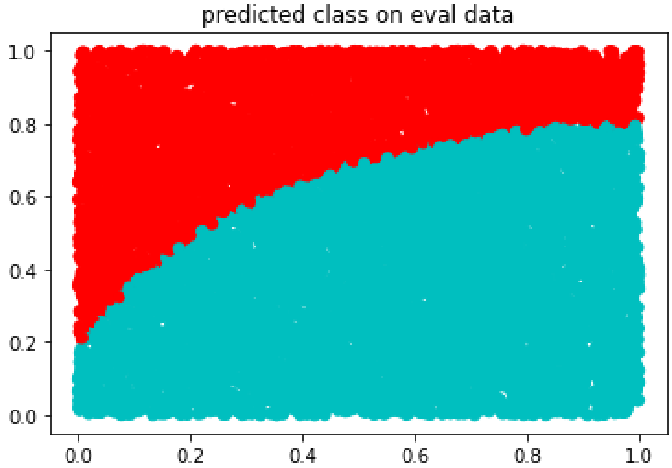

```
Errors: 1102
```

# 2b)

In [5]:

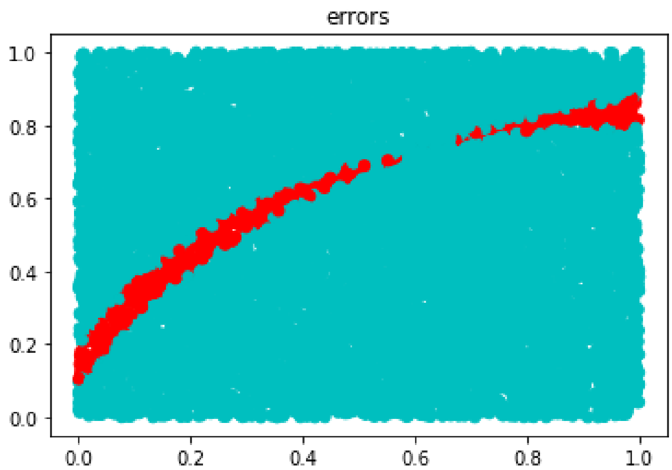
```
## Classifier 2
x_train_2 = np.hstack((x_train**2, x_train, np.ones((n_train,1)) ))
x_eval_2 = np.hstack((x_eval**2, x_eval, np.ones((n_eval,1)) ))

w_opt_2 = np.linalg.inv(x_train_2.transpose()@x_train_2)@x_train_2.transpose()@y_train
y_hat_2 = np.sign(x_eval_2@w_opt_2)

plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==-1 else 'r' for i in y_hat_2[:,0]]
)
plt.title('predicted class on eval data')
```

```
plt.show()
```




In [6]:

```
error_vec_2 = [0 if i[0]==i[1] else 1 for i in np.hstack((y_hat_2, y_eval))]
plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==0 else 'r' for i in error_vec_2])
plt.title('errors')
plt.show()

print('Error: '+ str(sum(error_vec_2)))
```

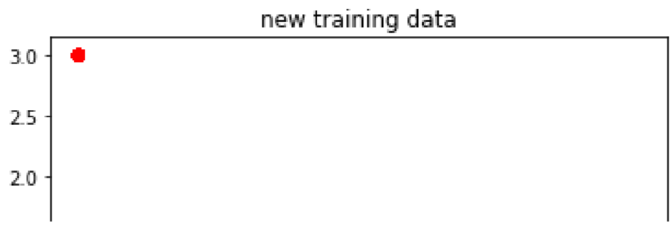


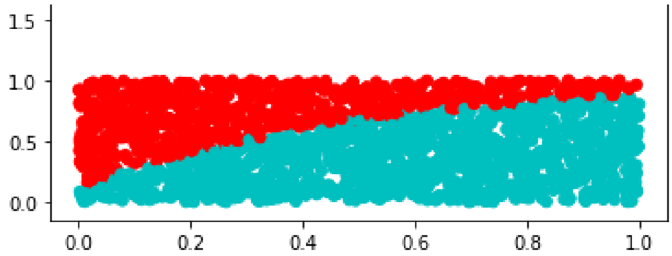

```
Error: 542
```

# 2c)

In [7]:

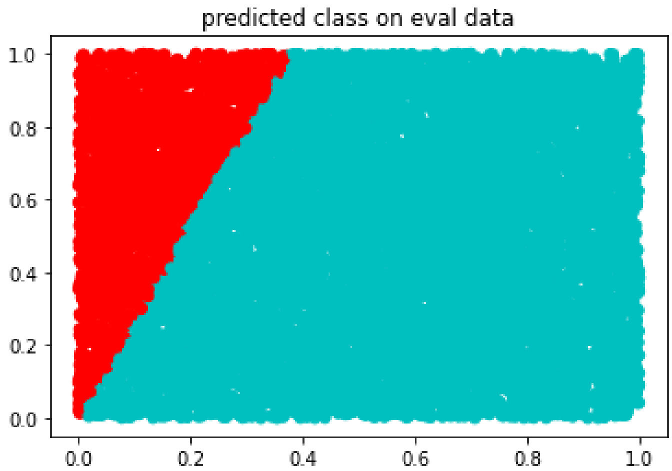
```
## create new, correctly labeled points
n_new = 1000 #number of new datapoints
x_train_new = np.hstack((np.zeros((n_new,1)), 3*np.ones((n_new,1))))
y_train_new = np.ones((n_new,1))

## add these to the training data
x_train_outlier = np.vstack((x_train,x_train_new))
y_train_outlier = np.vstack((y_train,y_train_new))
plt.scatter(x_train_outlier[:,0],x_train_outlier[:,1], color=['c' if i==-1 else 'r' for i
in y_train_outlier[:,0]])
plt.title('new training data')
plt.show()
```





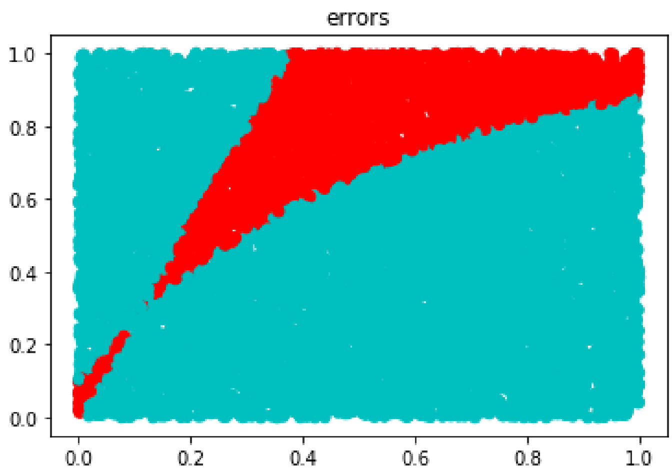

In [8]:

```
#train with new data
w_opt_outlier = np.linalg.inv(x_train_outlier.transpose()@x_train_outlier)@x_train_outlie
r.transpose()@y_train_outlier
y_hat_outlier = np.sign(x_eval@w_opt_outlier)

plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==-1 else 'r' for i in y_hat_outlier
[:,0]])
plt.title('predicted class on eval data')
plt.show()
```




In [9]:

```
error_vec = [0 if i[0]==i[1] else 1 for i in np.hstack((y_hat_outlier, y_eval))]
plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==0 else 'r' for i in error_vec])
plt.title('errors')
plt.show()

print('Errors: '+ str(sum(error_vec)))
```



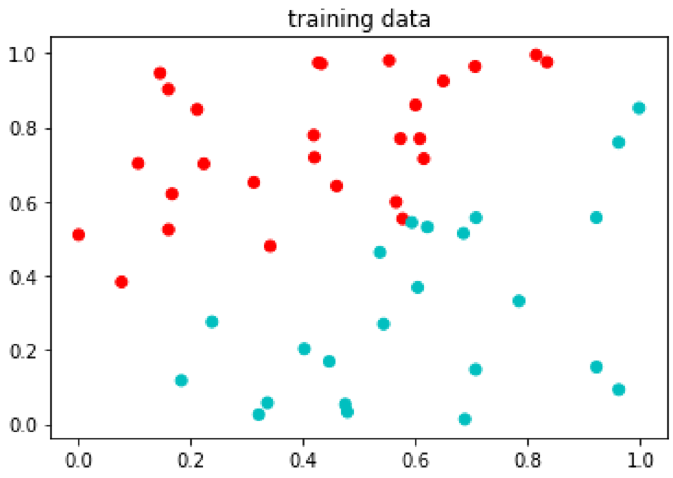

```
Errors: 2134
```

# 3a)

In [10]:

```
import numpy as np
```

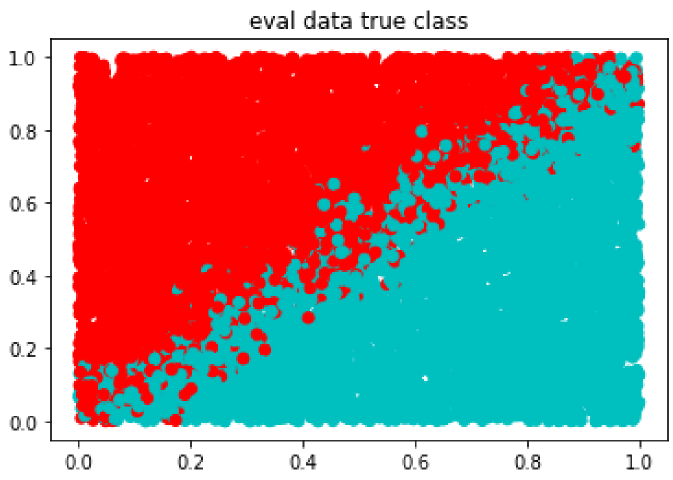
```
import numpy as np
from scipy.io import loadmat
import matplotlib.pyplot as plt

in_data = loadmat('overfitting_data.mat')
#print([key for key in in_data]) # -- use this line to see the keys in the dictionary dat
a structure

x_train = in_data['x_train']
x_eval = in_data['x_eval']
y_train = in_data['y_train']
y_eval = in_data['y_eval']

n_eval = np.size(y_eval)
n_train = np.size(y_train)

plt.scatter(x_train[:,0],x_train[:,1], color=['c' if i==-1 else 'r' for i in y_train[:,0
]])
plt.title('training data')
plt.show()
```




# 3b)

In [11]:

```
plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==-1 else 'r' for i in y_eval[:,0]])
plt.title('eval data true class')
plt.show()
```

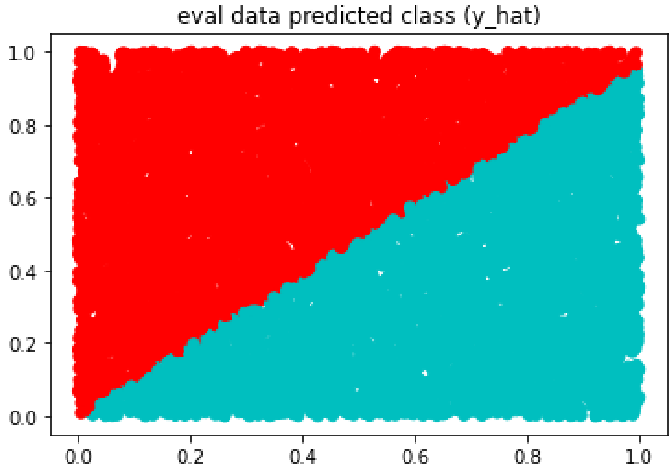



# 3c)

In [12]:

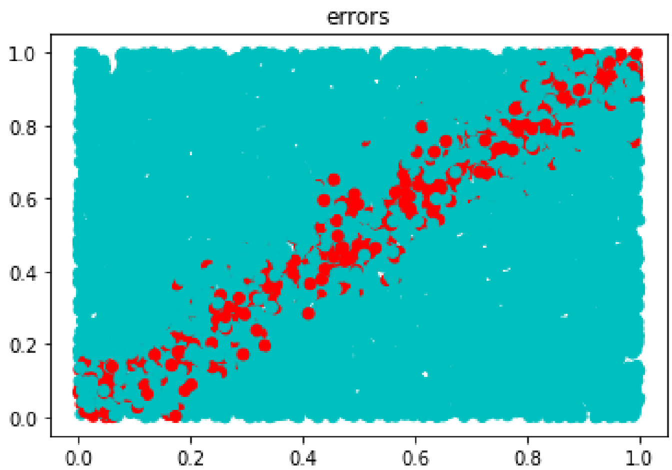
```
## Classifier 1
```

```python
# w = (X^T X)^(-1)X^T y
w_opt = np.linalg.inv(x_train.transpose()@x_train)@x_train.transpose()@y_train
y_hat = np.sign(x_eval@w_opt)

plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==-1 else 'r' for i in y_hat[:,0]])
plt.title('eval data predicted class (y_hat)')
plt.show()
```



In [13]:

```python
error_vec = [0 if i[0]==i[1] else 1 for i in np.hstack((y_hat, y_eval))]
plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==0 else 'r' for i in error_vec])
plt.title('errors')
plt.show()

print('Errors: '+ str(sum(error_vec)))
```



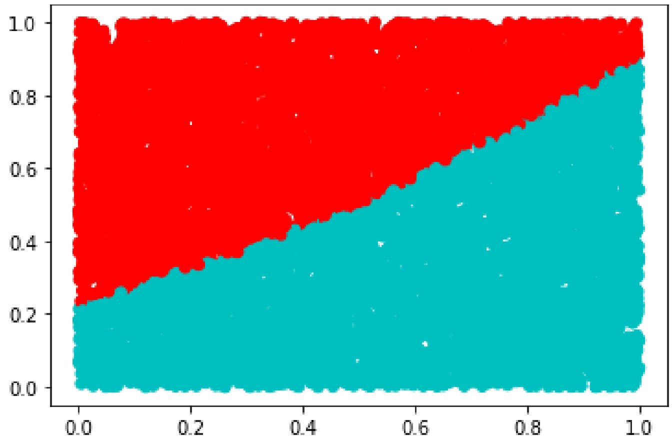
Errors: 759

# 3d)

In [14]:

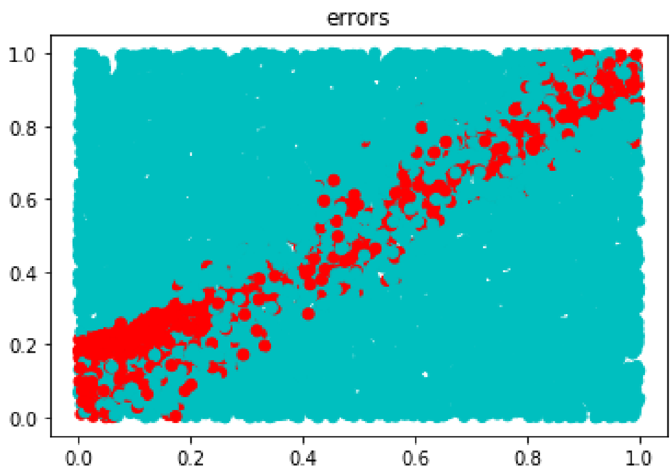
```python
## Classifier 2
x_train_2 = np.hstack((x_train**2, x_train, np.ones((n_train,1)) ))
x_eval_2 = np.hstack((x_eval**2,x_eval,  np.ones((n_eval,1)) ))


w_opt_2 = np.linalg.inv(x_train_2.transpose()@x_train_2)@x_train_2.transpose()@y_train
y_hat_2 = np.sign(x_eval_2@w_opt_2)

plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==-1 else 'r' for i in y_hat_2[:,0]]
)
plt.title('predicted class on eval data')
plt.show()
```

predicted class on eval data




In [15]:

```
error_vec_2 = [0 if i[0]==i[1] else 1 for i in np.hstack((y_hat_2, y_eval))]
plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==0 else 'r' for i in error_vec_2])
plt.title('errors')
plt.show()

print('Error: '+ str(sum(error_vec_2)))
```

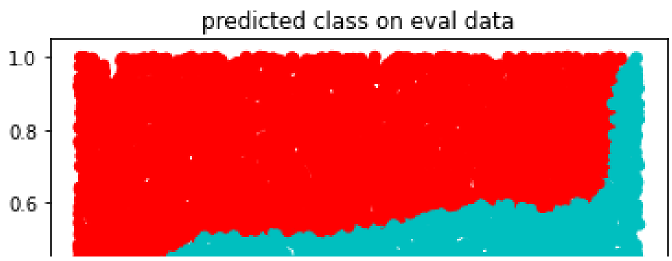


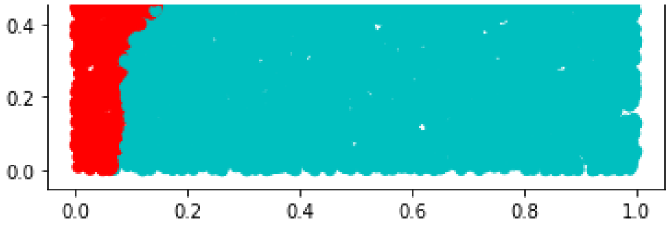

```
Error: 1066
```

# 3e)

In [16]:

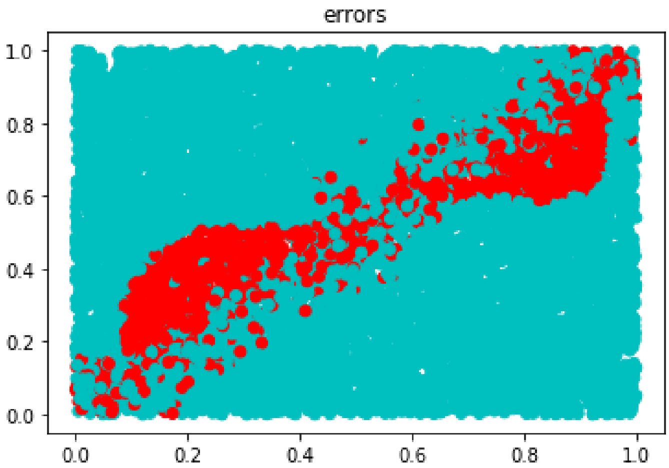
```
## Classifier 2
x_train_2 = np.hstack((x_train**6,x_train**5,x_train**4,x_train**3,x_train**2, x_train,
np.ones((n_train,1)) ))
x_eval_2 = np.hstack((x_eval**6,x_eval**5,x_eval**4,x_eval**3,x_eval**2,x_eval,  np.ones
((n_eval,1)) ))


w_opt_2 = np.linalg.inv(x_train_2.transpose()@x_train_2)@x_train_2.transpose()@y_train
y_hat_2 = np.sign(x_eval_2@w_opt_2)

plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==-1 else 'r' for i in y_hat_2[:,0]]
)
plt.title('predicted class on eval data')
plt.show()
```






In [17]:

```
error_vec_2 = [0 if i[0]==i[1] else 1 for i in np.hstack((y_hat_2, y_eval))]
plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==0 else 'r' for i in error_vec_2])
plt.title('errors')
plt.show()

print('Error: '+ str(sum(error_vec_2)))
```




Error: 1677

In [ ]: