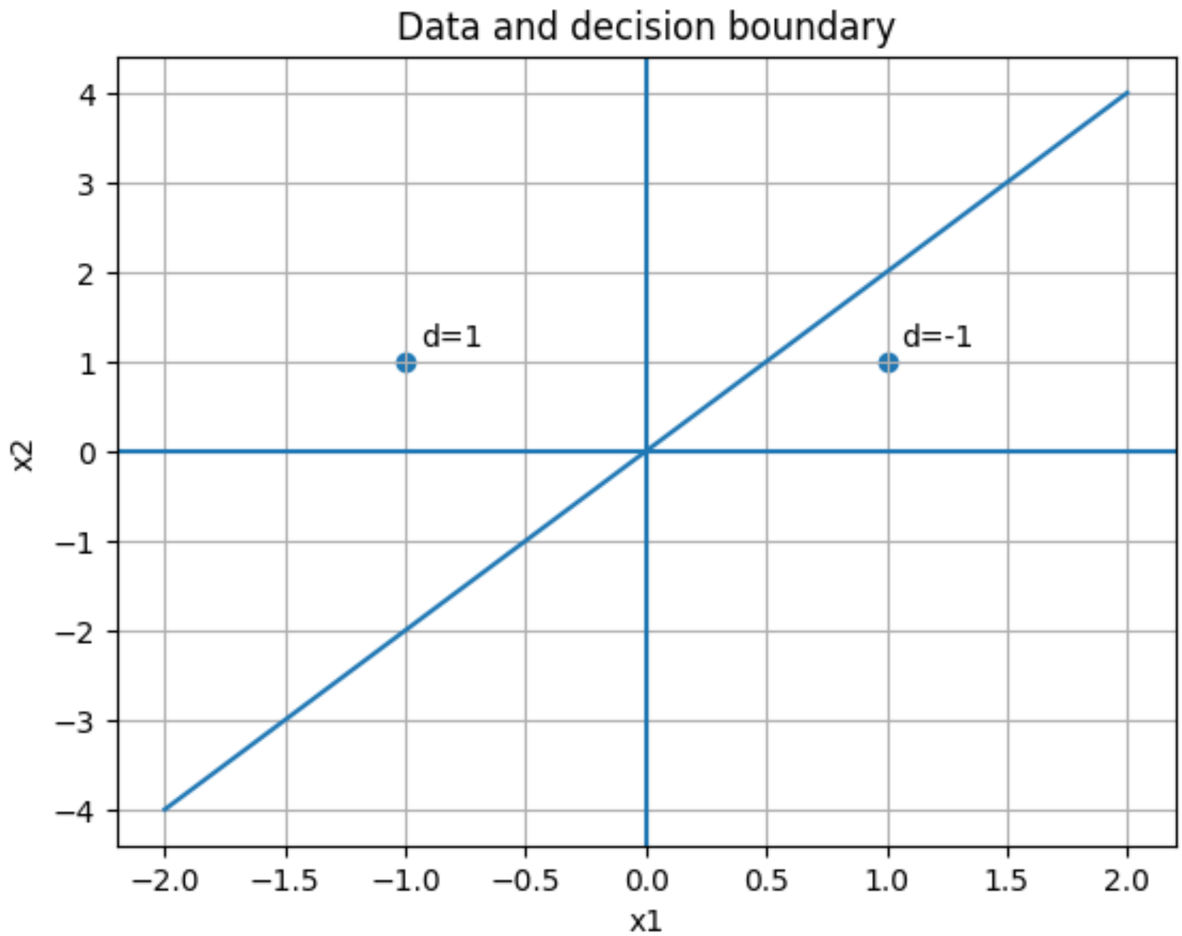
Question 1-a

In [2]:

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

# Data
x = np.array([[1, 1], [-1, 1]])
d = np.array([-1, 1])
w = np.array([-1, 0.5])

# Plot
plt.figure()
# Plot points
plt.scatter(x[:,0], x[:,1], marker='o')
for i, txt in enumerate(d):
    plt.annotate(f"d={txt}", (x[i,0], x[i,1]), textcoords="offset points", x

# Decision boundary: w[0]*x + w[1]*y = 0 => y = -w[0]/w[1] * x
xs = np.linspace(-2, 2, 100)
ys = -w[0]/w[1] * xs
plt.plot(xs, ys)

plt.axhline(0)
plt.axvline(0)
plt.xlabel('x1')
plt.ylabel('x2')
plt.title('Data and decision boundary')
plt.grid(True)
plt.show()
```



squared error loss:

$$w = \begin{bmatrix} -1 \\ 0.5 \end{bmatrix}$$

$$y_i = x_i^T w$$

$$loss = (d_i - y_i)^2$$

$$loss = (-1 - \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 0.5 \end{bmatrix})^2 + (1 - \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 0.5 \end{bmatrix})^2 = (-1 + 0.5)^2 + (1 - 1.5)^2 =$$

Question 1-b

hinge loss:

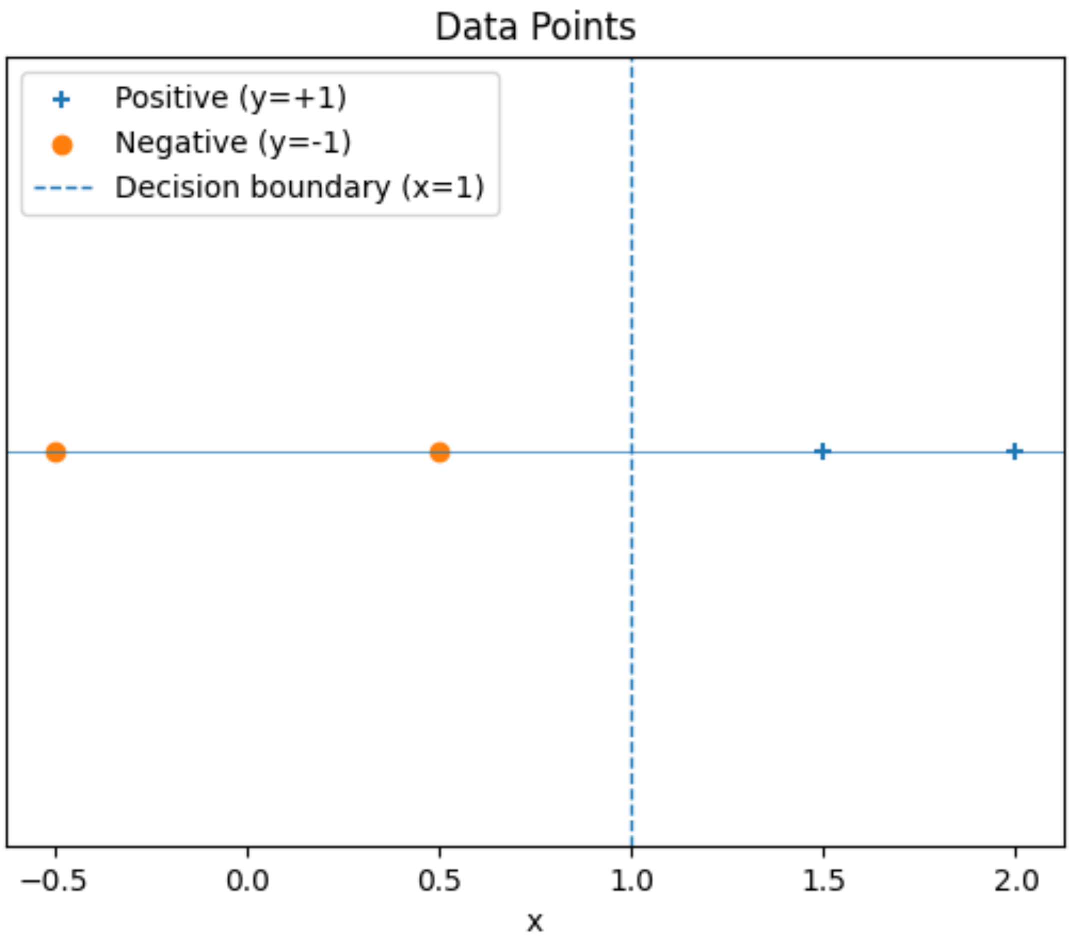
$$\ell(w, A, d) = \sum_{i=1}^{N} (1 - d_i x_i^T w)$$

$$w = \begin{bmatrix} -1 \\ 0.5 \end{bmatrix}$$

$$\ell(w, A, d) = (1 - (-1)(-0.5)) + (1 - (1)(1.5)) = 0$$

Question 2-a

In [17]:
```
import matplotlib.pyplot as plt
import numpy as np

# Data points
x = np.array([2, 1.5, 0.5, -0.5])
y = np.array([1, 1, -1, -1])

# Separate positive and negative
x_pos = x[y == 1]
y_pos = y[y == 1]
x_neg = x[y == -1]
y_neg = y[y == -1]

# Plot
plt.figure()
plt.scatter(x_pos, np.zeros_like(x_pos), marker='+', label='Positive (y=+1)'
plt.scatter(x_neg, np.zeros_like(x_neg), marker='o', label='Negative (y=-1)'
plt.axhline(0, linewidth=0.5)
plt.axvline(1, linestyle='--', linewidth=1, label='Decision boundary (x=1)')
plt.yticks([])
plt.xlabel('x')
plt.title('Data Points')
plt.legend()
plt.show()
```

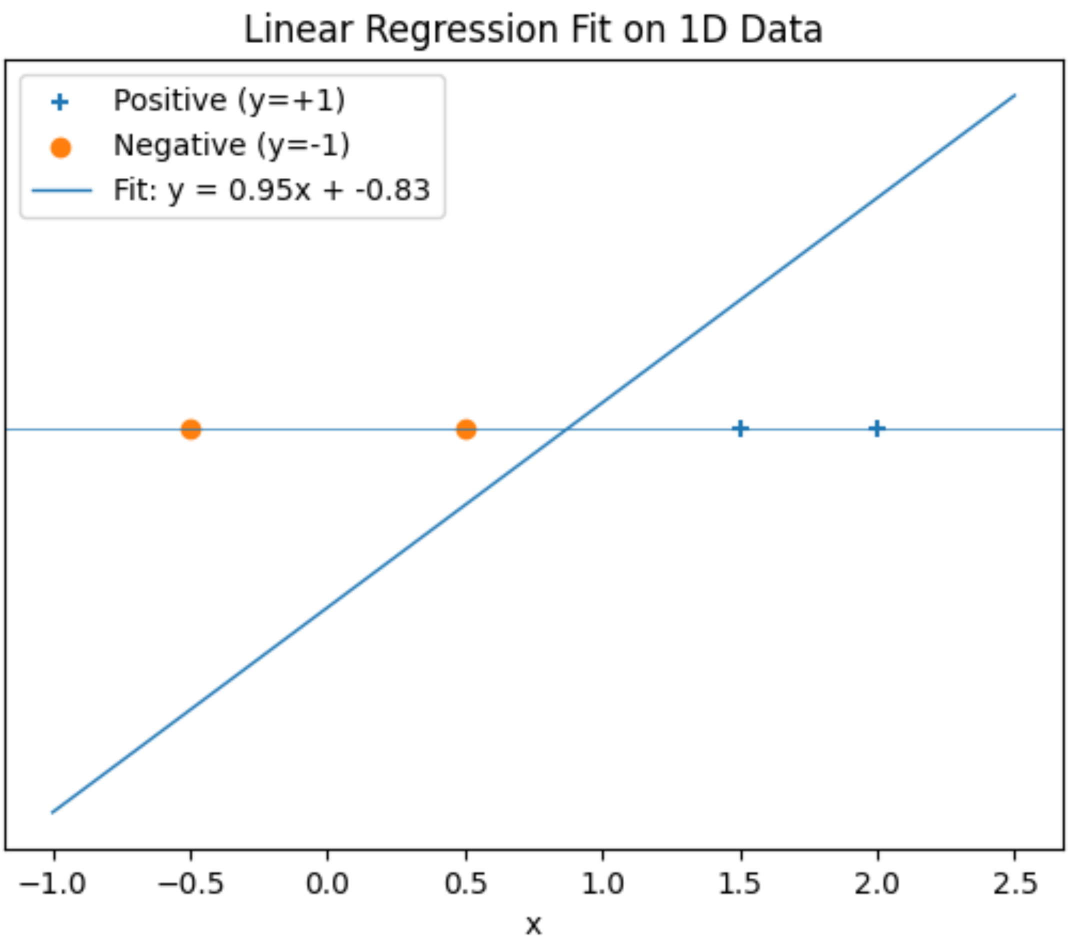


The max margin claassifier would be at $x = 1$

Question 2-b

In [19]:
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression

# Data
X = np.array([2, 1.5, 0.5, -0.5]).reshape(-1,1)
y = np.array([1, 1, -1, -1])

# Fit linear model
lr = LinearRegression().fit(X, y)
w, b = lr.coef_[0], lr.intercept_

# Prepare for plotting
x_vals = np.linspace(-1, 2.5, 100)
y_vals = w * x_vals + b

# Separate positive and negative points
x_pos = X[y == 1].flatten()
x_neg = X[y == -1].flatten()

# Plot
```

```
plt.figure()
plt.scatter(x_pos, np.zeros_like(x_pos), marker='+', label='Positive (y=+1)'
plt.scatter(x_neg, np.zeros_like(x_neg), marker='o', label='Negative (y=-1)'
plt.plot(x_vals, y_vals, linewidth=1, label=f'Fit: y = {w:.2f}x + {b:.2f}')
plt.axhline(0, linewidth=0.5)
plt.yticks([])
plt.xlabel('x')
plt.title('Linear Regression Fit on 1D Data')
plt.legend()
plt.show()
```

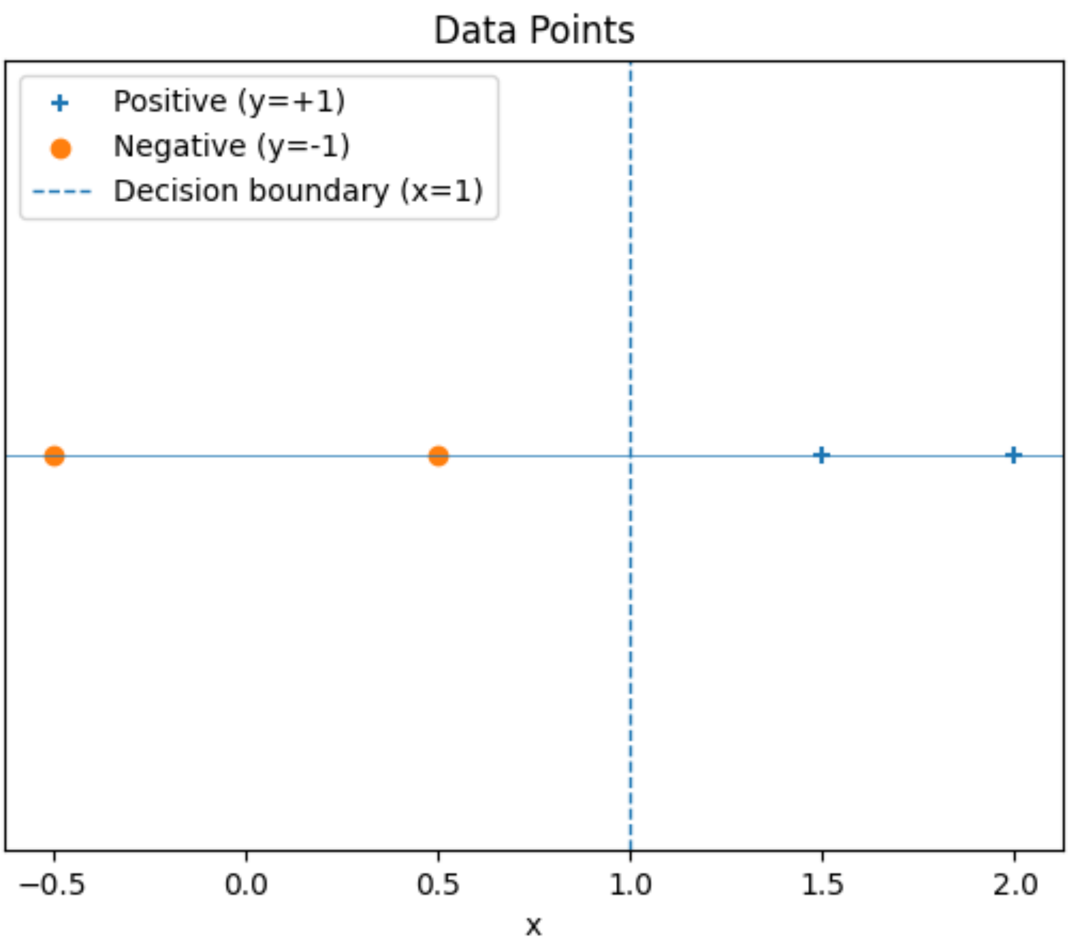


This model will make no errors

Question 2-c

In [21]:
```
import matplotlib.pyplot as plt
import numpy as np

# Data points
x = np.array([2, 1.5, 0.5, -0.5])
y = np.array([1, 1, -1, -1])

# Separate positive and negative
x_pos = x[y == 1]
y_pos = y[y == 1]
```

```python
x_neg = x[y == -1]
y_neg = y[y == -1]

# Plot
plt.figure()
plt.scatter(x_pos, np.zeros_like(x_pos), marker='+', label='Positive (y=+1)'
plt.scatter(x_neg, np.zeros_like(x_neg), marker='o', label='Negative (y=-1)'
plt.axhline(0, linewidth=0.5)
plt.axvline(1, linestyle='--', linewidth=1, label='Decision boundary (x=1)')
plt.yticks([])
plt.xlabel('x')
plt.title('Data Points')
plt.legend()
plt.show()
```

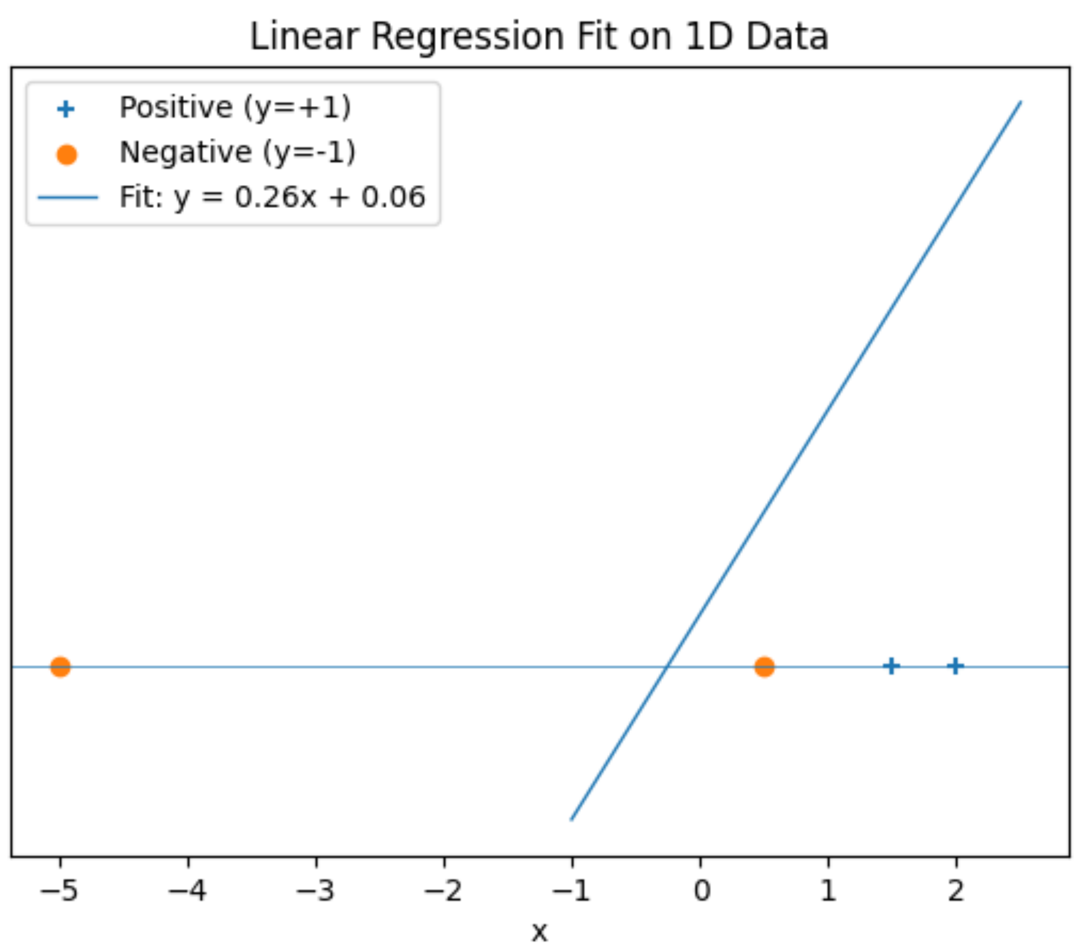


This classifier make no errors

Question 2-d

In [23]:
```python
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression

# Data
X = np.array([2, 1.5, 0.5, -5]).reshape(-1,1)
```

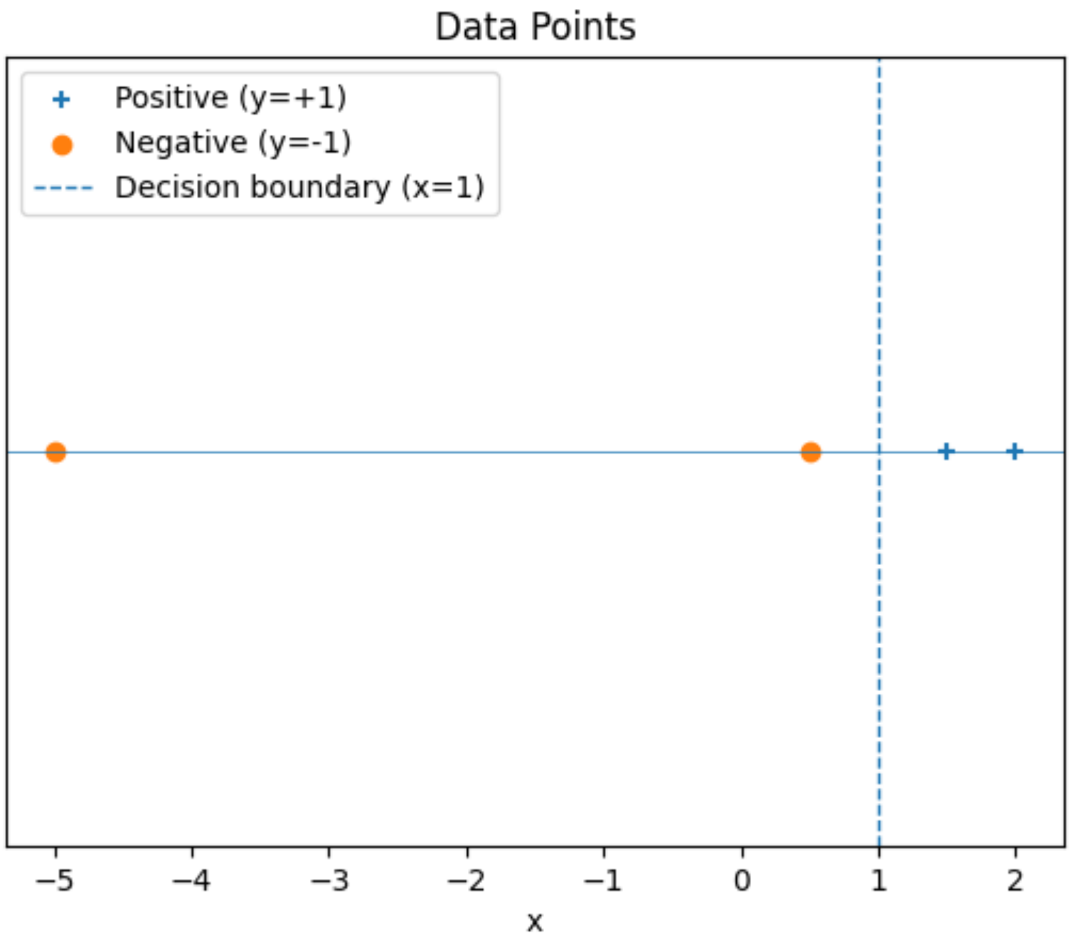
```
y = np.array([1, 1, -1, -1])

# Fit linear model
lr = LinearRegression().fit(X, y)
w, b = lr.coef_[0], lr.intercept_

# Prepare for plotting
x_vals = np.linspace(-1, 2.5, 100)
y_vals = w * x_vals + b

# Separate positive and negative points
x_pos = X[y == 1].flatten()
x_neg = X[y == -1].flatten()

# Plot
plt.figure()
plt.scatter(x_pos, np.zeros_like(x_pos), marker='+', label='Positive (y=+1)'
plt.scatter(x_neg, np.zeros_like(x_neg), marker='o', label='Negative (y=-1)'
plt.plot(x_vals, y_vals, linewidth=1, label=f'Fit: y = {w:.2f}x + {b:.2f}')
plt.axhline(0, linewidth=0.5)
plt.yticks([])
plt.xlabel('x')
plt.title('Linear Regression Fit on 1D Data')
plt.legend()
plt.show()
```



Yes there is error now

Question 2-e

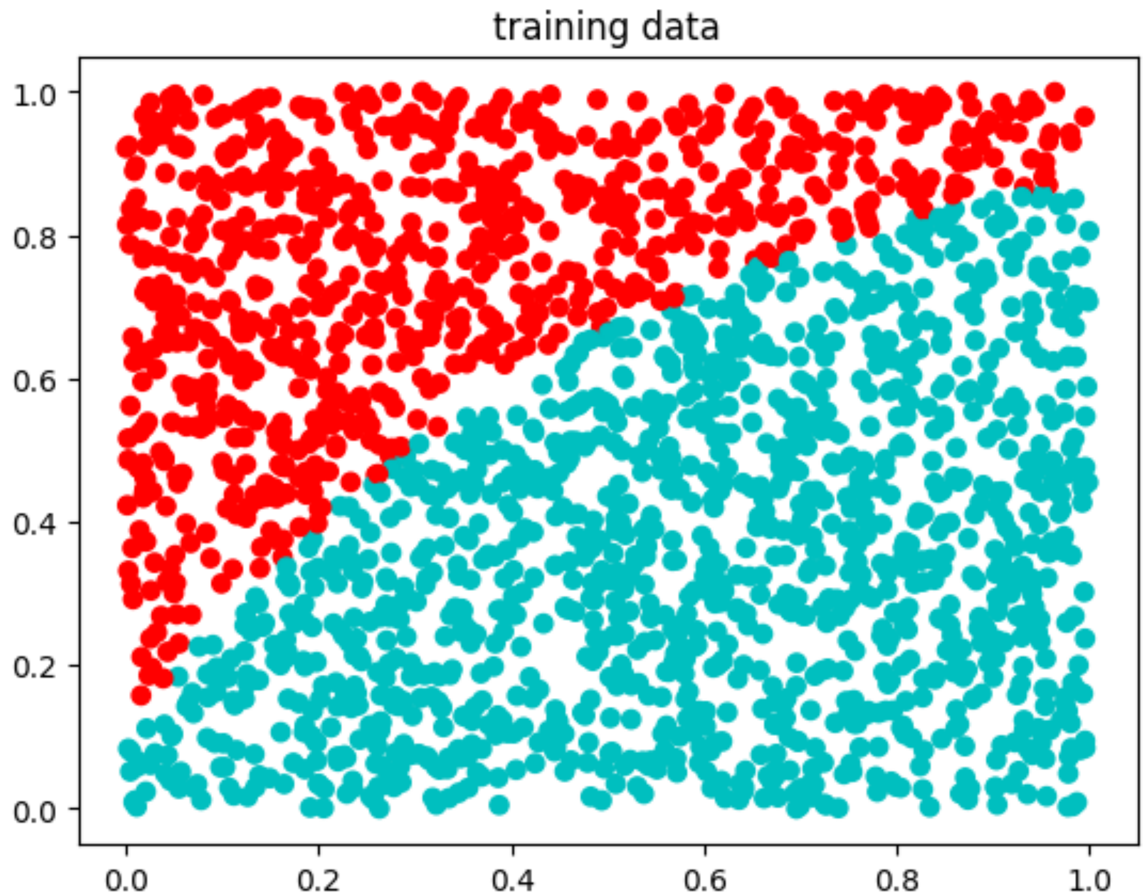
In [24]:
```
import matplotlib.pyplot as plt
import numpy as np

# Data points
x = np.array([2, 1.5, 0.5, -5])
y = np.array([1, 1, -1, -1])

# Separate positive and negative
x_pos = x[y == 1]
y_pos = y[y == 1]
x_neg = x[y == -1]
y_neg = y[y == -1]

# Plot
plt.figure()
plt.scatter(x_pos, np.zeros_like(x_pos), marker='+', label='Positive (y=+1)'
plt.scatter(x_neg, np.zeros_like(x_neg), marker='o', label='Negative (y=-1)'
plt.axhline(0, linewidth=0.5)
plt.axvline(1, linestyle='--', linewidth=1, label='Decision boundary (x=1)')
plt.yticks([])
plt.xlabel('x')
plt.title('Data Points')
plt.legend()
plt.show()
```



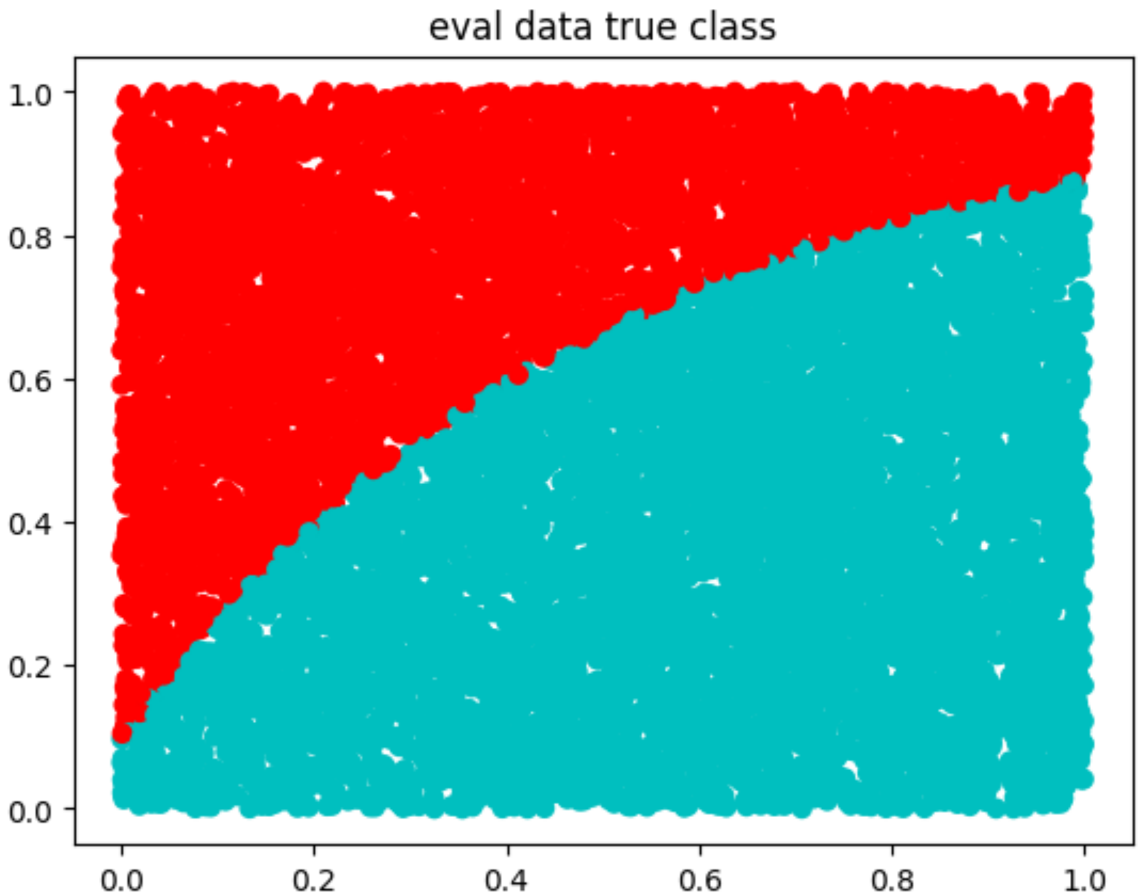
The hinge loss classifier doesnt change and still make no errors

Question 3-a

In [26]:
```
import numpy as np
from scipy.io import loadmat
import matplotlib.pyplot as plt
from sklearn.svm import LinearSVC

in_data = loadmat('classifier_data.mat')

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
plt.title('training data')
plt.show()
```

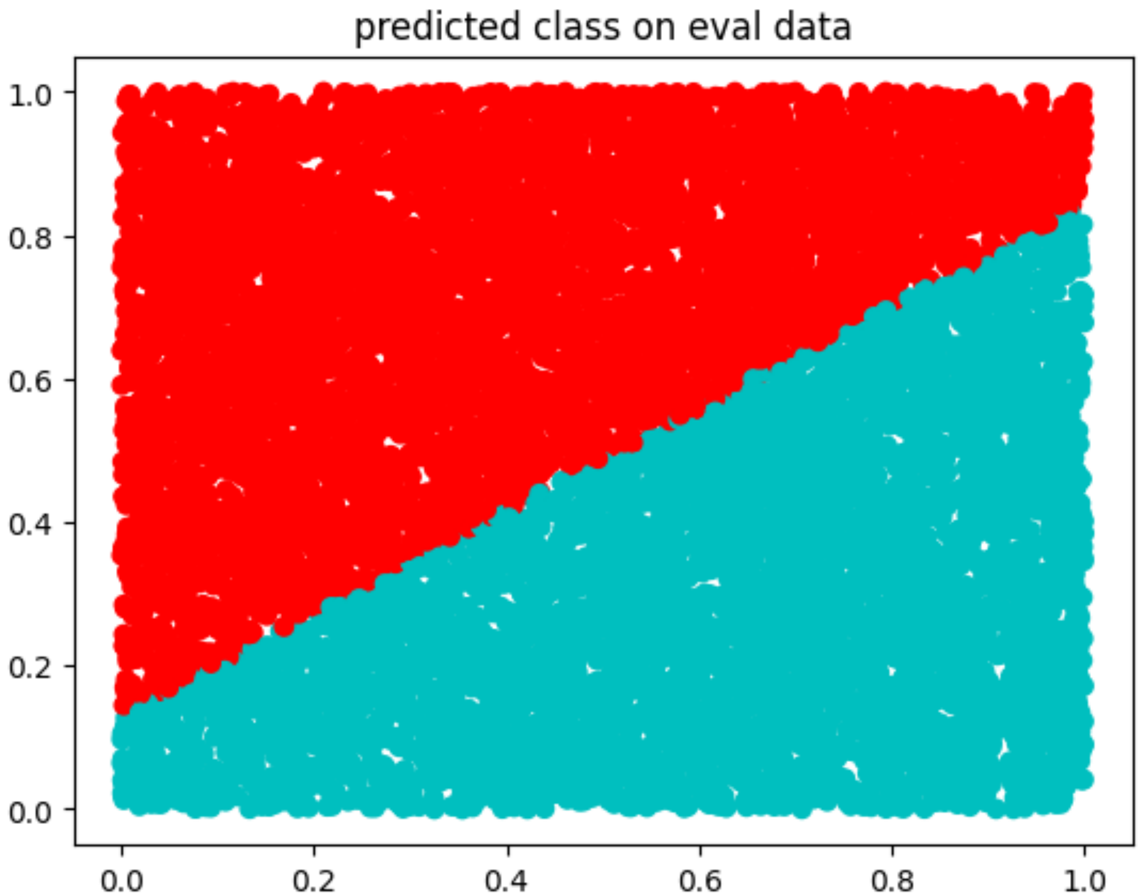


In [27]:
```
plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==-1 else 'r' for i in y
plt.title('eval data true class')
plt.show()
```

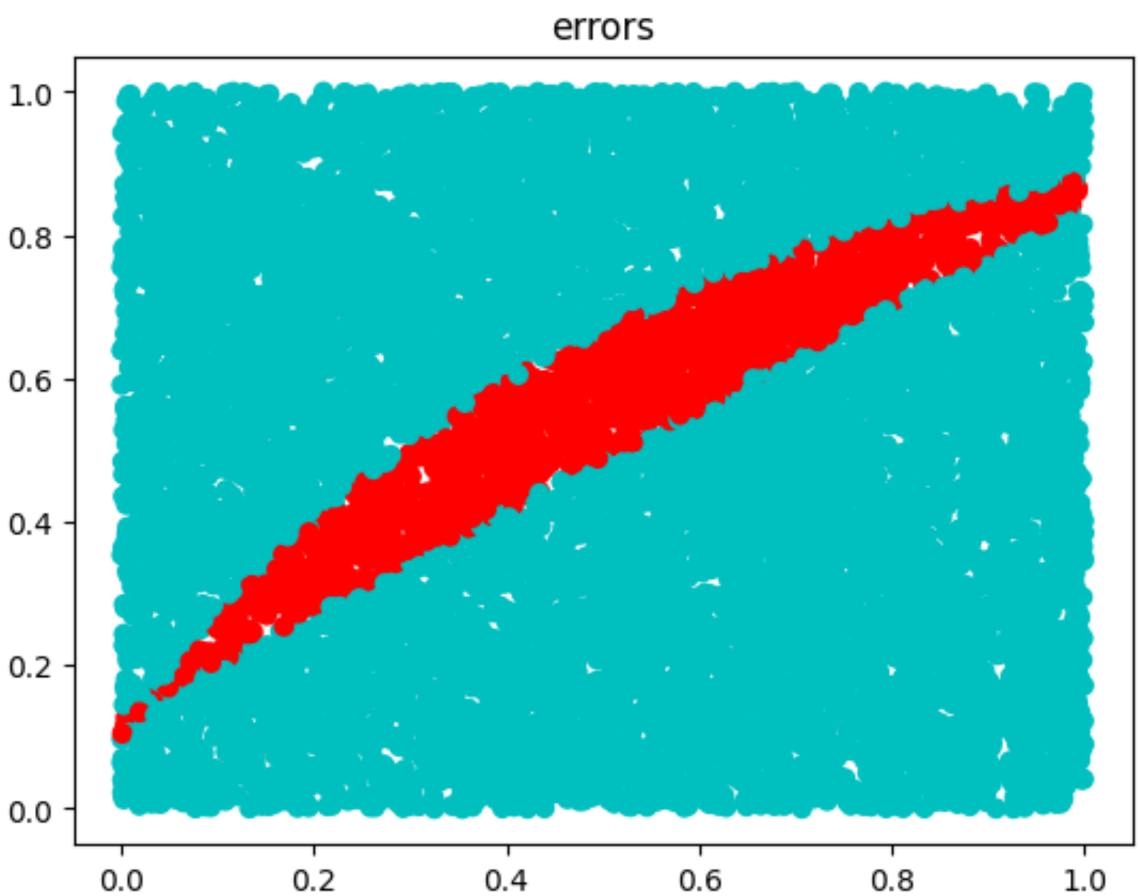


In [28]:

```
## Classifier 1
x_train_1 = np.hstack(( x_train, np.ones((n_train,1)) ))
x_eval_1 = np.hstack(( x_eval, np.ones((n_eval,1)) ))

# Train classifier using linear SVM from SK Learn library
clf = LinearSVC(random_state=0, tol=1e-8)
clf.fit(x_train_1, np.squeeze(y_train))
w_opt = clf.coef_.transpose()

#uncomment this line to use least squares classifier
#w_opt = np.linalg.inv(x_train_1.T@x_train_1)@x_train_1.T@y_train

y_hat_outlier = np.sign(x_eval_1@w_opt)
plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==-1 else 'r' for i in y
plt.title('predicted class on eval data')
plt.show()
```



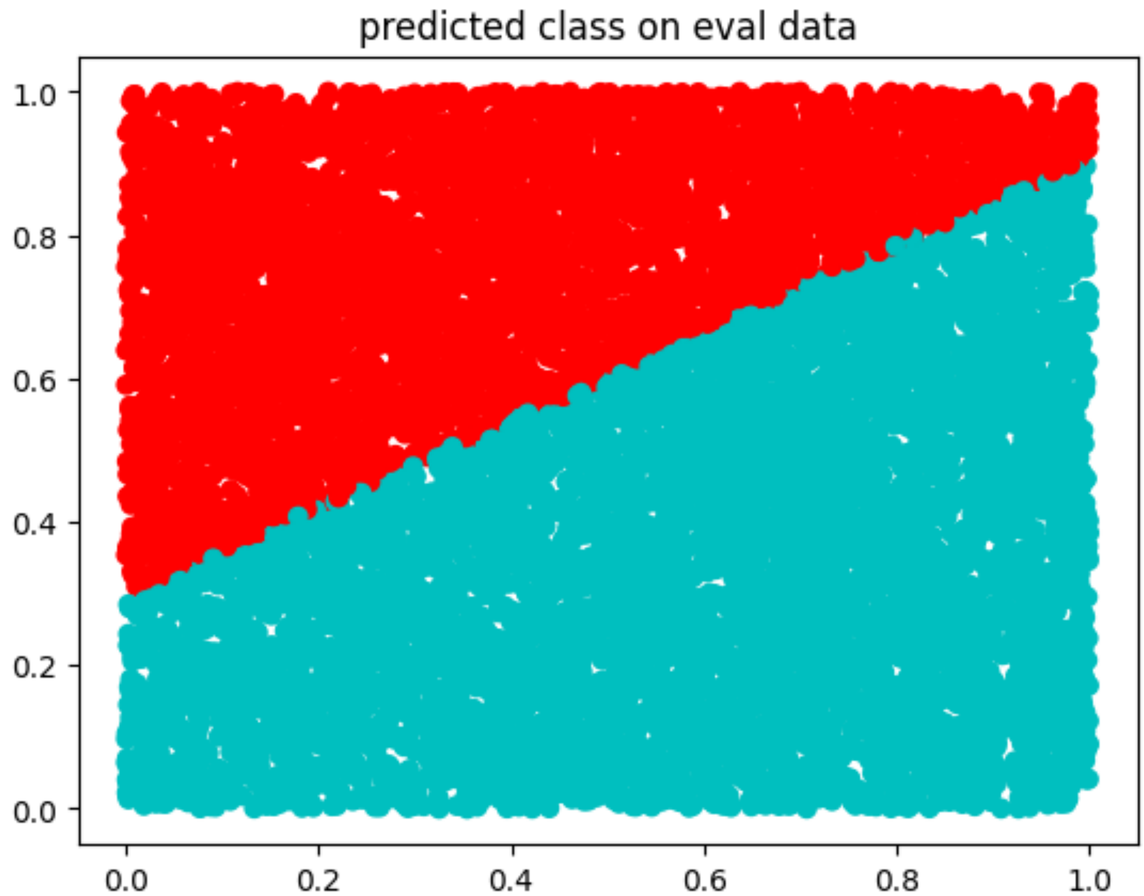
In [29]:
```
error_vec = [0 if i[0]==i[1] else 1 for i in np.hstack((y_hat_outlier, y_eva
plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==0 else 'r' for i in er
plt.title('errors')
plt.show()

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

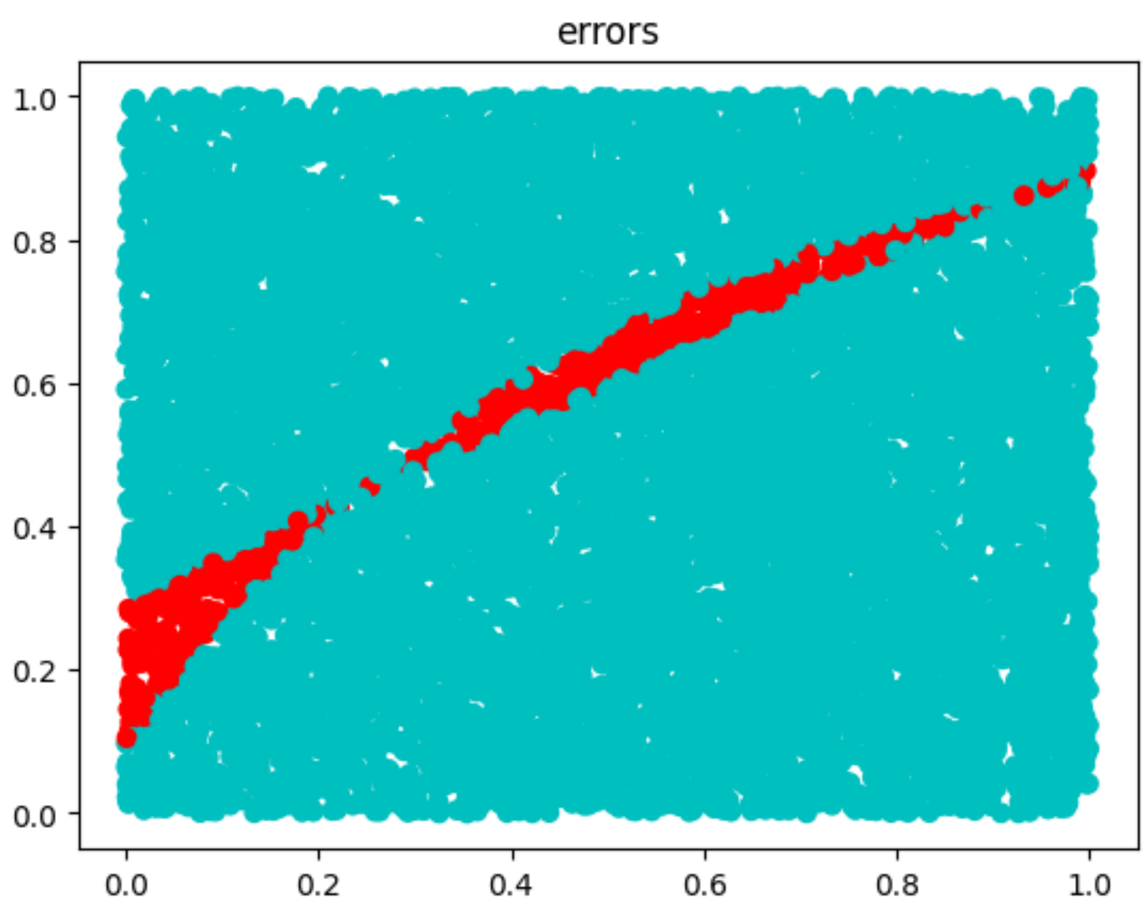



```
Errors: 1213
```

Question 3-b

```
In [30]: ## Classifier 1
x_train_1 = np.hstack(( x_train, np.ones((n_train,1)) ))
x_eval_1 = np.hstack(( x_eval, np.ones((n_eval,1)) ))

# Train classifier using linear SVM from SK Learn library
clf = LinearSVC(random_state=0, tol=1e-8)
clf.fit(x_train_1, np.squeeze(y_train))
# w_opt = clf.coef_.transpose()

#uncomment this line to use least squares classifier
w_opt = np.linalg.inv(x_train_1.T@x_train_1)@x_train_1.T@y_train

y_hat_outlier = np.sign(x_eval_1@w_opt)
plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==-1 else 'r' for i in y
plt.title('predicted class on eval data')
plt.show()

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

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

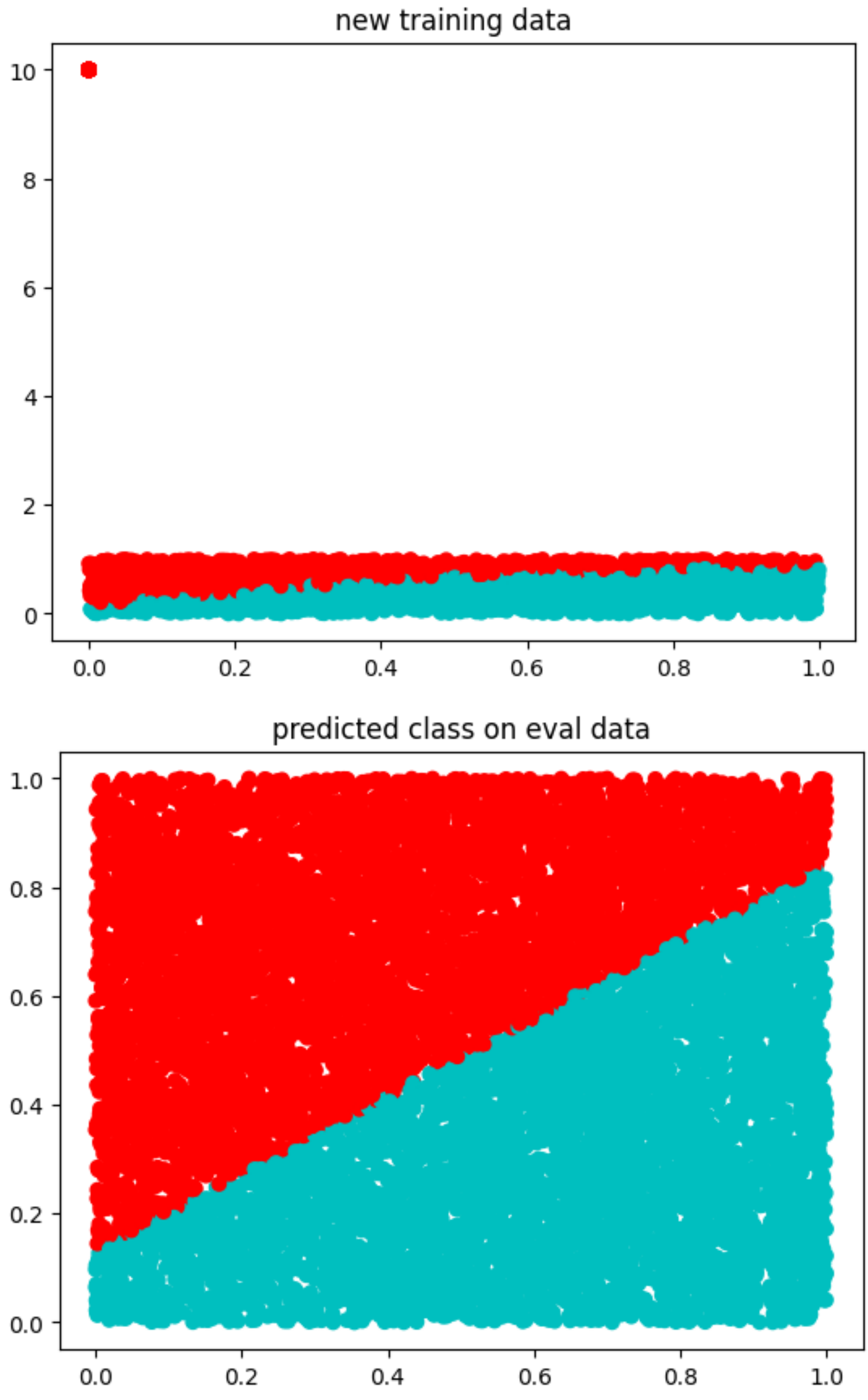

predicted class on eval data
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

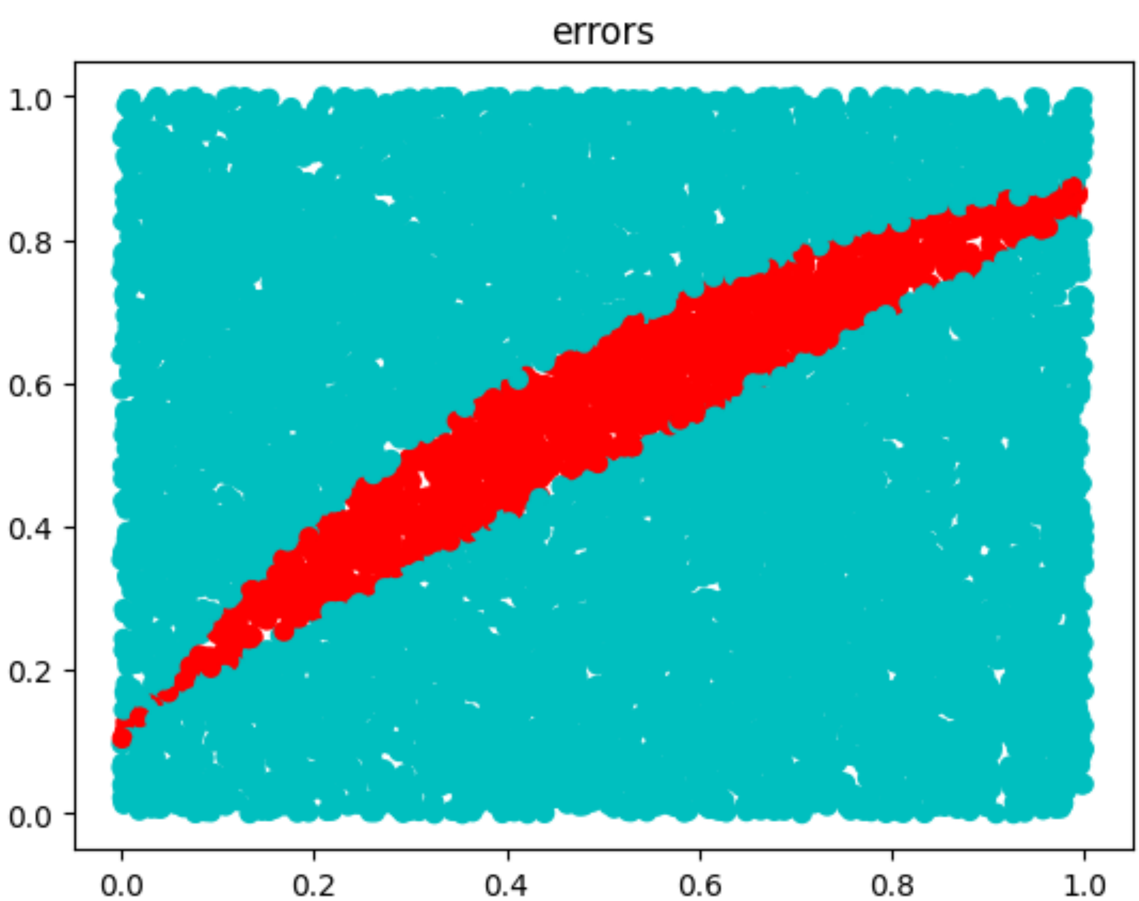

errors
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

```
Errors: 495
```

Question 2-c

In [34]:
```
## create new, correctly labeled points
n_new = 1000 #number of new datapoints
x_train_new = np.hstack((np.zeros((n_new,1)), 10*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 e
plt.title('new training data')
plt.show()

x_train_outlier_1 = np.hstack((x_train_outlier, np.ones((n_train+n_new,1)) )
x_eval_1 = np.hstack((x_eval, np.ones((n_eval,1)) ))

#Train classifier using off the shelf SVM from sklearn
clf = LinearSVC(random_state=0, tol=1e-5)
clf.fit(x_train_outlier_1, np.squeeze(y_train_outlier))
w_opt_outlier = clf.coef_.transpose()

#uncomment this line to use least squares classifier
# w_opt_outlier = np.linalg.inv(x_train_outlier_1.T@x_train_outlier_1)@x_tra

y_hat_outlier = np.sign(x_eval_1@w_opt_outlier)
plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==-1 else 'r' for i in y
plt.title('predicted class on eval data')
plt.show()

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

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


new training data
10
8
6
4
2
0
0.0
0.2
0.4
0.6
0.8
1.0
predicted class on eval data
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0



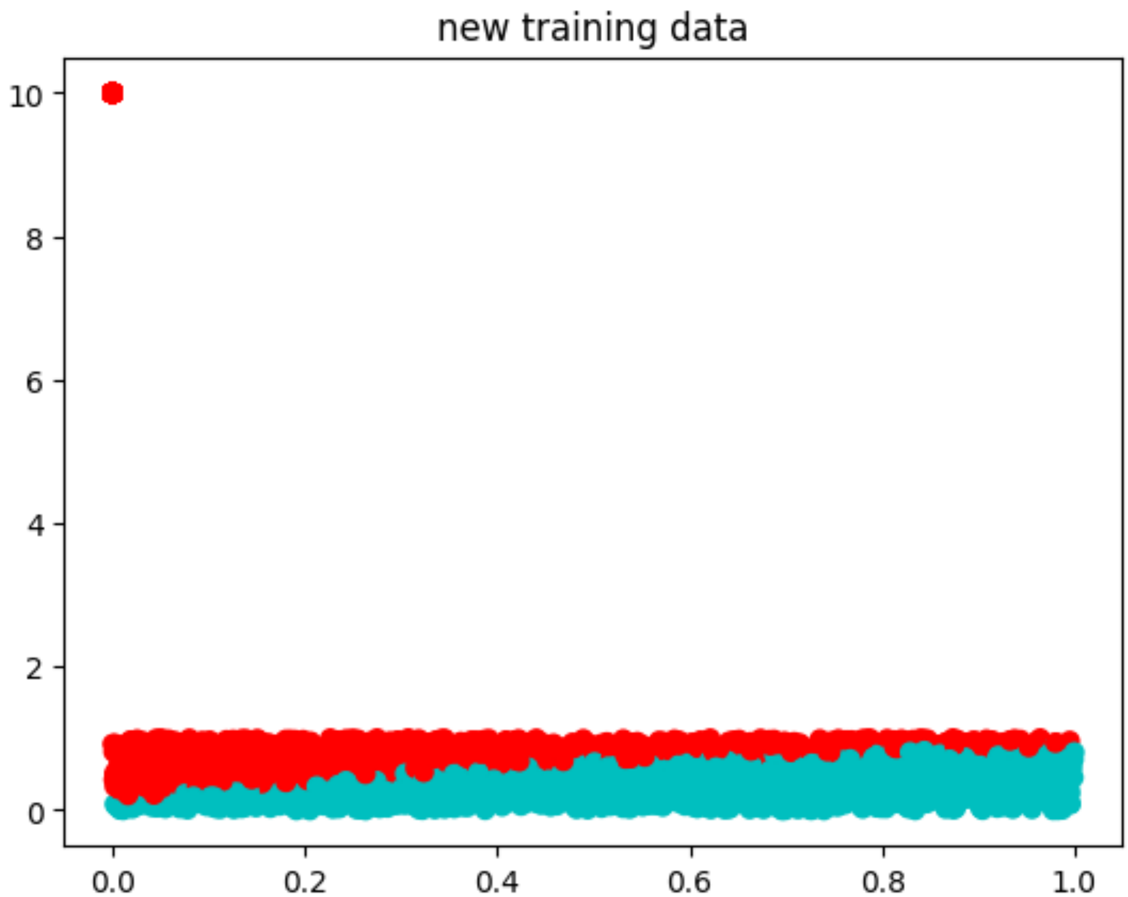

```
Errors: 1213
```

The boundary stays the same and the error doesnt change

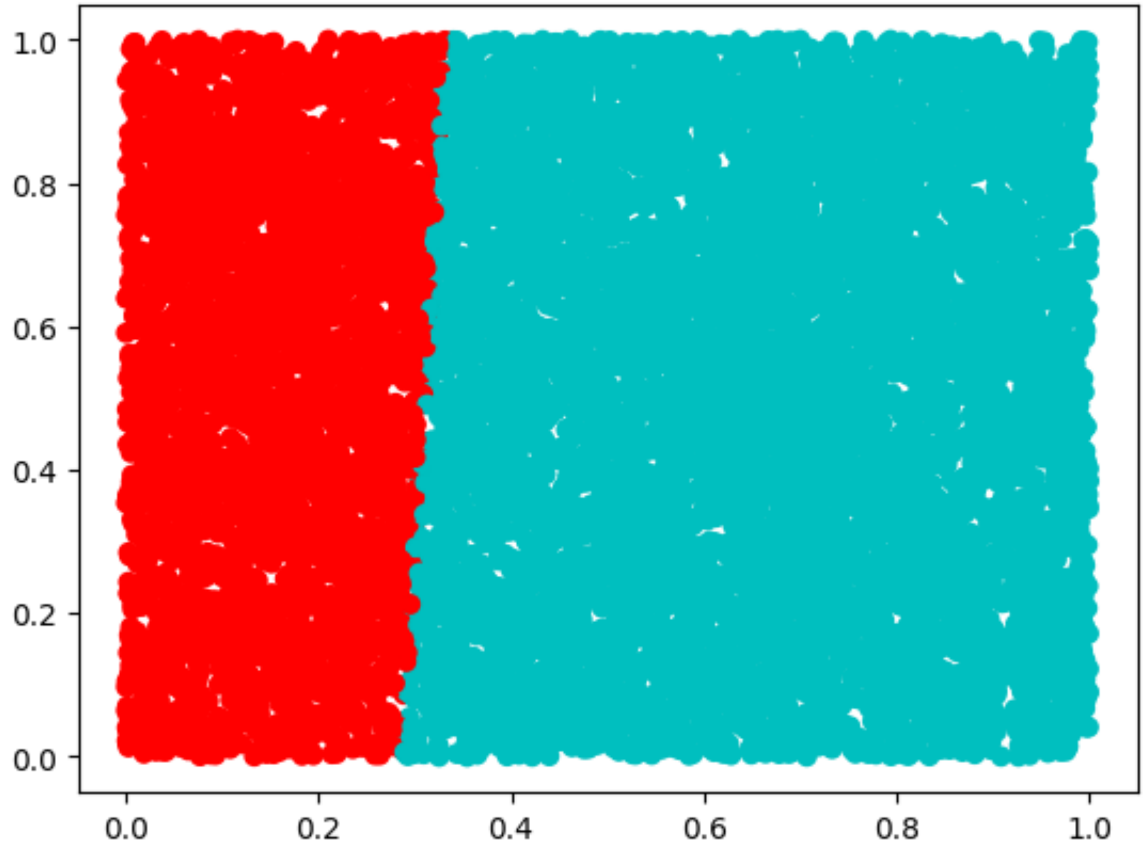
Question 3-d

```
In [35]: ## create new, correctly labeled points
n_new = 1000 #number of new datapoints
x_train_new = np.hstack((np.zeros((n_new,1)), 10*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 e
plt.title('new training data')
plt.show()

x_train_outlier_1 = np.hstack((x_train_outlier, np.ones((n_train+n_new,1)) )
x_eval_1 = np.hstack((x_eval, np.ones((n_eval,1)) ))

#Train classifier using off the shelf SVM from sklearn
clf = LinearSVC(random_state=0, tol=1e-5)
clf.fit(x_train_outlier_1, np.squeeze(y_train_outlier))
# w_opt_outlier = clf.coef_.transpose()

#uncomment this line to use least squares classifier
```

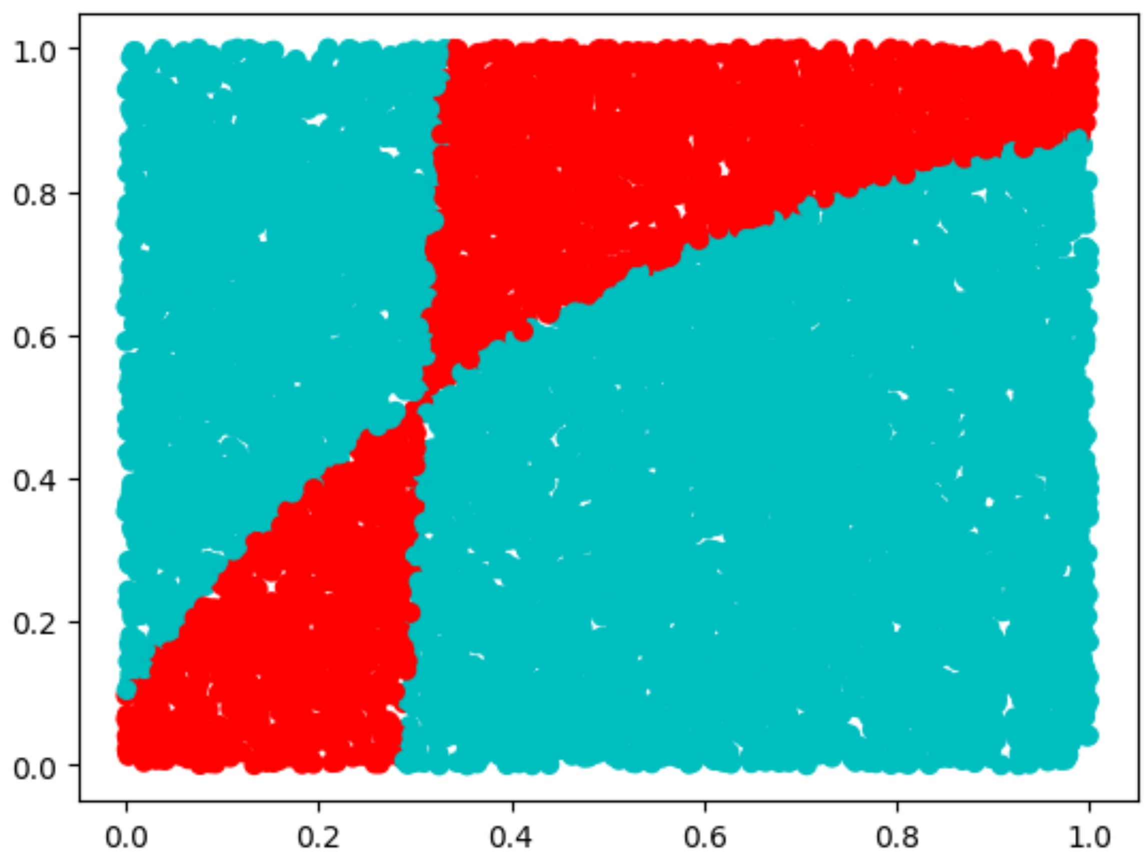
```
w_opt_outlier = np.linalg.inv(x_train_outlier_1.T@x_train_outlier_1)@x_trair

y_hat_outlier = np.sign(x_eval_1@w_opt_outlier)
plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==-1 else 'r' for i in y
plt.title('predicted class on eval data')
plt.show()

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

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



predicted class on eval data



errors




```
Errors: 2668
```

The linear classifier boundary tries to acount for the outlier and fails to classify the data as well as before

The model tries to account for the outlier in the squared loss function