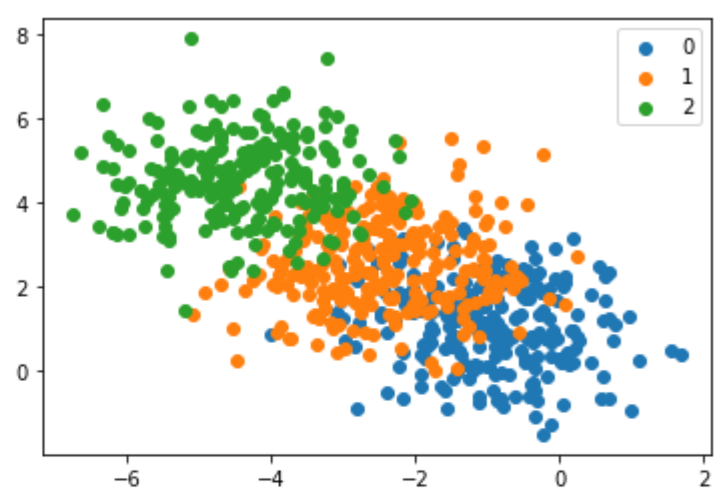
In [4]:
```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
```

In [5]:
```python
# 1.1
miu_zero = [-1, 1]
miu_one = [-2.5, 2.5]
miu_two = [-4.5, 4.5]

sigma = [1, 1]
samples = np.zeros([700, 3])
for i in range(700):
    miu_index = 0
    rand_index = np.random.rand()
    if rand_index >= 2/3:
        miu_index = 2
        sample = np.random.normal(miu_two, sigma)
    elif 1/3 < rand_index < 2/3:
        miu_index = 1
        sample = np.random.normal(miu_one, sigma)
    else:
        sample = np.random.normal(miu_zero, sigma)

    samples[i] = [sample[0], sample[1], miu_index]
```

In [6]:
```python
# 1.2
data = pd.DataFrame(samples, columns = ['x','y','label'])

miu_zero_df = data[data['label'] == 0]
miu_one_df = data[data['label'] == 1]
miu_two_df = data[data['label'] == 2]
```

In [7]:
```python
plt.scatter(x=miu_zero_df['x'], y=miu_zero_df['y'], label='0')
plt.scatter(x=miu_one_df['x'], y=miu_one_df['y'], label='1')
plt.scatter(x=miu_two_df['x'], y=miu_two_df['y'], label='2')
plt.legend()
```

Out[7]:
```
<matplotlib.legend.Legend at 0x7ffbcbe81250>
```



In [8]:
```
# 1.3
test_samples = np.zeros([300, 3])
for i in range(300):
    miu_index = 0
    rand_index = np.random.rand()
    if rand_index >= 2/3:
        miu_index = 2
        sample = np.random.normal(miu_two, sigma)
    elif 1/3 < rand_index < 2/3:
        miu_index = 1
        sample = np.random.normal(miu_one, sigma)
    else:
        sample = np.random.normal(miu_zero, sigma)

    test_samples[i] = [sample[0], sample[1], miu_index]
test_data = pd.DataFrame(test_samples, columns=['x', 'y', 'label'])
```

In [9]:
```
miu_zero_df = test_data[test_data['label'] == 0]
miu_one_df = test_data[test_data['label'] == 1]
miu_two_df = test_data[test_data['label'] == 2]

plt.scatter(x=miu_zero_df['x'], y=miu_zero_df['y'], label='0')
plt.scatter(x=miu_one_df['x'], y=miu_one_df['y'], label='1')
plt.scatter(x=miu_two_df['x'], y=miu_two_df['y'], label='2')
plt.legend()
plt.title('test data')

plt.show()
```

4.

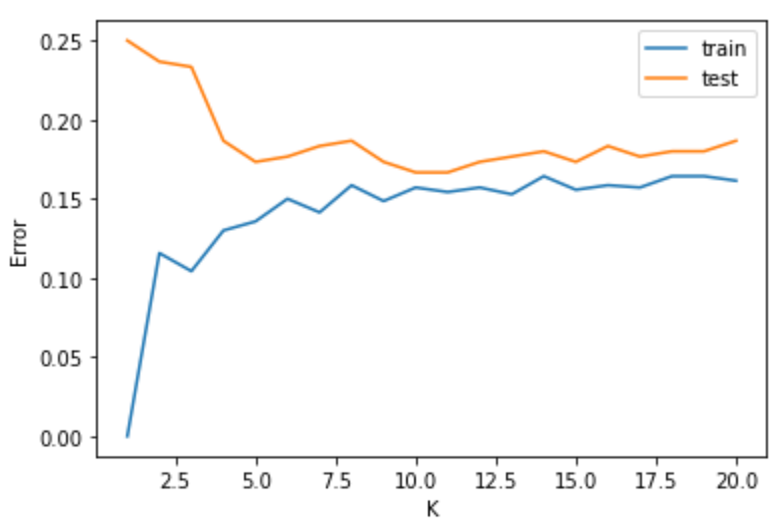
In [10]:
```python
# 1.4
k = 1
x_cols = ['x', 'y']
y_cols = 'label'
model = KNeighborsClassifier(n_neighbors=k)
model.fit(X=data[x_cols], y=data[y_cols])
y_train_pred = model.predict(X=data[x_cols])
y_test_pred = model.predict(X=test_data[x_cols])

y_train_true = data[y_cols].values
y_test_true = test_data[y_cols].values
train_error = np.sum(y_train_pred != y_train_true) / len(y_train_true)
test_error = np.sum(y_test_pred != y_test_true) / len(y_test_true)
print(train_error)
print(test_error)
```

```
0.0
0.25
```

expected to increase accuracy with increase in K in the test set.

In [11]:
```python
# 1.5
errors = []
for i in range(1, 21):
    k = i
    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(X=data[x_cols], y=data[y_cols])
    y_train_pred = model.predict(X=data[x_cols])
    y_test_pred = model.predict(X=test_data[x_cols])

    y_train_true = data[y_cols].values
    y_test_true = test_data[y_cols].values
    train_accuracy = np.sum(y_train_pred == y_train_true) / len(y_train_true)
    test_accuracy = np.sum(y_test_pred == y_test_true) / len(y_test_true)
    errors.append([1- train_accuracy, 1- test_accuracy])
```

In [12]:
```python
plt.plot( range(1,21), np.transpose(errors)[0], label='train')
plt.plot( range(1,21), np.transpose(errors)[1], label='test')
plt.xlabel("K")
plt.ylabel("Error")
plt.legend()
plt.show()
```



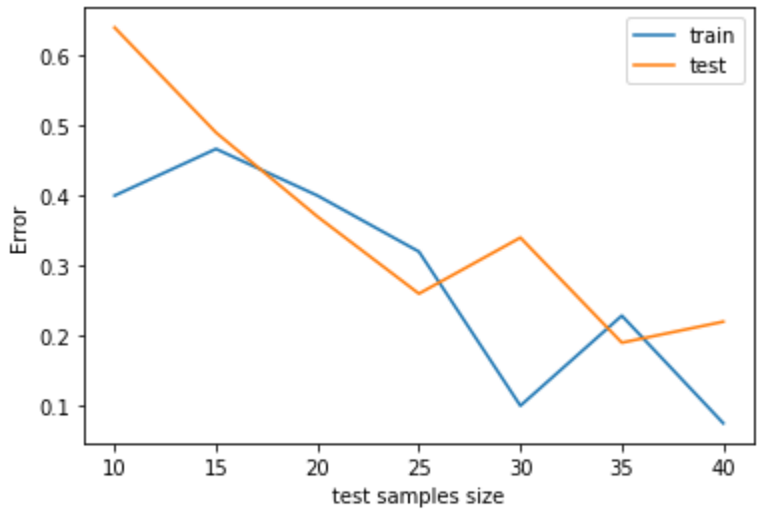
the test error decreased with k increase, matched expectations the test not always decrease with k , for example at the graph we got we have the error increase from k=10 to k=~13

6. we expect the error to decrease with m_train-i increase:

In [13]:
```
# 1.6
def build_train(train_size):
    samples = np.zeros([train_size, 3])
    for i in range(train_size):
        miu_index = 0
        rand_index = np.random.rand()
        if rand_index >= 2 / 3:
            miu_index = 2
            sample = np.random.normal(miu_two, sigma)
        elif 1 / 3 < rand_index < 2 / 3:
            miu_index = 1
            sample = np.random.normal(miu_one, sigma)
        else:
            sample = np.random.normal(miu_zero, sigma)

        samples[i] = [sample[0], sample[1], miu_index]

    data = pd.DataFrame(samples, columns=['x', 'y', 'label'])
    return data
```

In [14]:
```
error = []

new_test = build_train(100)
for i in range(10, 41, 5):
    k = 10
    data = build_train(i)
    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(X=data[x_cols], y=data[y_cols])
    y_train_pred = model.predict(X=data[x_cols])
    y_test_pred = model.predict(X=new_test[x_cols])

    y_train_true = data[y_cols].values
    y_test_true = new_test[y_cols].values
    train_error = np.sum(y_train_pred != y_train_true) / len(y_train_true)
    test_error = np.sum(y_test_pred != y_test_true) / len(y_test_true)
    error.append([train_error, test_error])
```

In [15]:
```
plt.plot( range(10, 41, 5), np.transpose(error)[0], label='train')
plt.plot( range(10, 41, 5), np.transpose(error)[1], label='test')
plt.xlabel("test samples size")
plt.ylabel("Error")
plt.legend()
plt.show()
```



as we expected the error was overall decreasing, but the graph was erratic due to low train size.

1.7 and 1.8 are in the PDF

שאלה 1 (7 – 8):

1.7
Yes, the plots change between the trials. Not always meet our expectations (from step 6) at every trial but the general trend is descending.

1.8
We will calculate y by the weighted score of the neighbors in the following way:
p = (x_i, y_i),

$$y = argmax_{label} \sum_{p \in N_k(x)} \frac{w(x, x_i)}{\sum_{j=1}^{k} w(x, x_i)} * 1[y_i = label],$$

$$w(x, x_i) = \frac{1}{d(x, x_i)}$$

We are doing $\frac{1}{d}$ and not just d to give the closer points a heavier weight, and the far points less weight.

שאלה 3

1.

$$P_w(Y_i = k|x_i) = \left(\frac{e^{w_k^T x_i}}{\sum_{j=1}^{K} e^{w_j^T x_i}}\right) = p_k =>$$

$$for\ w_1, w_2\ lets\ find\ w\ such\ that\ p_1 = p, p_2 = 1 - p$$

$$p_1 = \left(\frac{e^{w_1^T x_i}}{e^{w_1^T x_i} + e^{w_2^T x_i}}\right) p_2 = \left(\frac{e^{w_2^T x_i}}{e^{w_1^T x_i} + e^{w_2^T x_i}}\right)$$

נגדיר ש $w = w_1 - w_2$

$$p = \left(\frac{e^{w^T x_i}}{1 + e^{w^T x_i}}\right) = \left(\frac{e^{(w_1 - w_2)^T x_i}}{1 + e^{(w_1 - w_2)^T x_i}}\right) = \left(\frac{\frac{e^{w_1^T x_i}}{e^{w_2^T x_i}}}{\frac{e^{w_2^T x_i}}{e^{w_2^T x_i}} + \frac{e^{w_1^T x_i}}{e^{w_2^T x_i}}}\right) = \frac{e^{w_1^T x_i}}{e^{w_2^T x_i}} * \frac{e^{w_2^T x_i}}{\left(e^{w_2^T x_i} + e^{w_1^T x_i}\right)}$$

$$= \frac{e^{w_1^T x_i}}{\left(e^{w_2^T x_i} + e^{w_1^T x_i}\right)} = p_1$$

$$1 - p_1 = 1 - \frac{e^{w_1^T x_i}}{\left(e^{w_2^T x_i} + e^{w_1^T x_i}\right)} = \frac{\left(e^{w_2^T x_i} + e^{w_1^T x_i}\right)}{\left(e^{w_2^T x_i} + e^{w_1^T x_i}\right)} - \frac{e^{w_1^T x_i}}{\left(e^{w_2^T x_i} + e^{w_1^T x_i}\right)} = \frac{e^{w_2^T x_i}}{\left(e^{w_2^T x_i} + e^{w_1^T x_i}\right)} = p_2$$

נמשיך ונראה כי זה נכון עבור כל w, ועבורw_1, w_2 מסוימים.
נגדיר $w = w_1, w_2 = 0$ ונקבל:

$$p_1 = \left(\frac{e^{w^T x_i}}{e^{w^T x_i} + 1}\right) => p_1 = p$$

$$p_2 = \left(\frac{1}{e^{w^T x_i} + 1}\right) = \frac{1 + e^{w^T x_i} - e^{w^T x_i}}{e^{w^T x_i} + 1} = \frac{e^{w^T x_i} + 1}{e^{w^T x_i} + 1} - \frac{e^{w^T x_i}}{e^{w^T x_i} + 1} = 1 - \frac{e^{w^T x_i}}{e^{w^T x_i} + 1} = 1 - p_1$$

.2

כדי לפתור את $argmax_w$ נציב את הפיתוח מסעיף קודם ונגזור לפי w_k לכל K מתאים:
ניזכר כי ראינו בהרצאה ש:

$$L_s(w) = \frac{1}{m}\sum_{i=1}^{m} -\log\left(P_w(Y_i = y_i|\boldsymbol{x_i})\right) = \frac{1}{m}\sum_{i=1}^{m}\left(\log\left(\sum_k e^{w_k^T x_i}\right) - w_{y_i}^T x_i\right)$$

ולכן, נגדיר $f_w(w) = \sum_{i=1}^{m} \log\left(P_w(Y_i = y_i|\boldsymbol{x_i})\right)$ ו-:

$$f_s(w) = \sum_{i=1}^{m} \log\left(P_w(Y_i = y_i|\boldsymbol{x_i})\right) = -m * L_s(w) = \sum_{i=1}^{m}\left(w_{y_i}^T x_i - \log\left(\sum_k e^{w_k^T x_i}\right)\right)$$

נגזור על ידי מה שראינו בתרגול, ונשווה ל-0:

$$\frac{\partial f_s(w)}{\partial w_k} = \sum_{i=1}^{m} 1[\mathrm{y_i} = \mathrm{k}]\mathrm{x_i} - \sum_{i=1}^{m}\left(\frac{e^{w_k^T x_i}}{\sum_{j=1}^{K} e^{w_j^T x_i}}\right) x_i = 0 =>$$

$$\boxed{\sum_{i=1}^{m} \mathbf{1}[\mathbf{y_i} = \mathbf{k}]\mathbf{x_i} = \sum_{i=1}^{m}\left(\frac{e^{w_k^T x_i}}{\sum_{j=1}^{K} e^{w_j^T x_i}}\right)\boldsymbol{x_i}}$$

.3

א. יש 3 וקטורי משקל w כיוון שיש 3 classes – וקטור מגדיר לכל מישור הפרדה לכל class.

ב. מכיוון שלוקטור x_i מוסיפים עוד רכיב של ה bias.
ג. אנו יודעים כי:

$$P_w(Y_i = k|x_i) = \frac{e^{w_k^T x_i}}{\sum_{j=1}^{K} e^{w_j^T x_i}}$$

ניזכר כי עלינו להוסיף עוד כניסה לכל ווקטור x_i שבה יהיה 1, בגלל הbias.
אנו מניחים כי לכל ווקטור w, הרכיב הראשון הוא זה המתאים לbias. נחשב (שימו לב כי נסווג אח"כ):

$\underline{\boldsymbol{x_1}}$:

$$\boldsymbol{p_{11}} = \frac{e^{w_1^T x_1}}{\sum_{j=1}^{K} e^{w_j^T x_1}} = \frac{e^{21.5}}{e^{21.5}+e^{-9.5}+e^{-12}} = \mathbf{1}$$

$$\boldsymbol{p_{12}} = \frac{e^{w_2^T x_1}}{\sum_{j=1}^{K} e^{w_j^T x_1}} = \frac{e^{-9.5}}{e^{21.5}+e^{-9.5}+e^{-12}} = \mathbf{0.0344 * 10^{-12} \approx 0}$$

$$\boldsymbol{p_{13}} = \frac{e^{w_3^T x_1}}{\sum_{j=1}^{K} e^{w_j^T x_1}} = \frac{e^{-12}}{e^{21.5}+e^{-9.5}+e^{-12}} = \mathbf{0.002525 * 10^{-12} \approx 0}$$

$\underline{\boldsymbol{x_2}}$:

$$\boldsymbol{p_{21}} = \frac{e^{w_1^T x_2}}{\sum_{j=1}^{K} e^{w_j^T x_2}} = \frac{e^{-11}}{e^{-11}+e^{8}+e^{3}} \approx \mathbf{0}$$

$$\boldsymbol{p_{22}} = \frac{e^{w_2^T x_2}}{\sum_{j=1}^{K} e^{w_j^T x_2}} = \frac{e^{8}}{e^{-11}+e^{8}+e^{3}} = \mathbf{0.9933071435}$$

$$\boldsymbol{p_{23}} = \frac{e^{w_3^T x_2}}{\sum_{j=1}^{K} e^{w_j^T x_2}} = \frac{e^{3}}{e^{-11}+e^{8}+e^{3}} \approx \mathbf{0}$$

$\underline{\boldsymbol{x_3}}$:

$$\boldsymbol{p_{31}} = \frac{e^{w_1^T x_3}}{\sum_{j=1}^{K} e^{w_j^T x_3}} = \frac{e^{-14}}{e^{-14}+e^{2}+e^{12}} \approx \mathbf{0}$$

$$\boldsymbol{p_{32}} = \frac{e^{w_2^T x_3}}{\sum_{j=1}^{K} e^{w_j^T x_3}} = \frac{e^{2}}{e^{-14}+e^{2}+e^{12}} \approx \mathbf{0}$$

$$\boldsymbol{p_{33}} = \frac{e^{w_3^T x_3}}{\sum_{j=1}^{K} e^{w_j^T x_3}} = \frac{e^{12}}{e^{-14}+e^{2}+e^{12}} = \mathbf{0.9999546021}$$

כלומר, על פי מה שקיבלנו נראה כי:

$$\boldsymbol{y_1 = 1}$$
$$\boldsymbol{y_2 = 2}$$
$$\boldsymbol{y_3 = 3}$$