1.

max $1 - \sum^{K} \mu_K^2$

s.t $\sum^{K} \mu_K = 1$

$L = 1 - \sum^{K} \mu_K^2 + \lambda\left(\sum^{K} \mu_K - 1\right)$

$\forall k$ $\frac{\partial L}{\partial \mu_k} = -2\mu_k + \lambda\mu_k \Rightarrow \mu_k = \frac{\lambda}{2}$

$\sum^{K} \mu_k = 1 \Rightarrow K \cdot \frac{\lambda}{2} = 1 \Rightarrow \lambda = \frac{2}{K}.$

$\mu_K^* = \frac{1}{K}$

max impurity $= 1 - K \cdot \frac{1}{K^2} = 1 - \frac{1}{K}$

2.

$\mu_+\left(1 - 2(\mu_+ - \mu_-) + (\mu_+ - \mu_-)^2\right)$

$+ \mu_-\left(1 - 2(\mu_+ - \mu_-) + (\mu_+ - \mu_-)^2\right)$

$= 1 + (\mu_+ - \mu_-)^2 - 2\mu_+^2 - 2\mu_-^2 + 4\mu_+\mu_-$

$= 1 - \mu_+^2 - \mu_-^2 \; 2\mu_+\mu_- \neq r\left(1 - \mu_+^2 - \mu_-^3\right)$

$\downarrow r \in \mathbb{R}.$

$\Rightarrow$ not scaled gini impurity.

3.

$$\lim_{N \to \infty} \left(1-\frac{1}{N}\right)^{pN} = \left[\left(1-\frac{1}{N}\right)^{N}\right]^{p} \approx e^{-p}$$

↳ 抽出 N' 個後，未被抽到的比例

$e^{-p} \cdot N$ is the number of unsample

4.

K個 classification.

至少要 $\frac{K+1}{2}$ 個都犯錯 G 才會出錯

在極端的情況下，將所有的錯誤 $\sum e_k$

集中在 $\frac{K+1}{2}$ 個分類器上，會使 $E_{out}(G)$ 放大.

此時 $E_{out}(G) \leq \sum^{K} e_k \Big/ \frac{K+1}{2} = \frac{2}{K+1} \sum^{K} e_k$

5. By Lecture 11 p.17.

α_1 is optimal η by g_1-transformed LinearReg

which $\min\limits_{\eta} \frac{1}{N} \sum \left((y_n - \underbrace{s_n}_{0}) - \eta \underbrace{g_1(x_n)}_{11.26} \right)^2$

By Formula of OLS.

$$\alpha_1 = \eta = \frac{\sum^{N} g_1(x_n)(y_n - \overset{0}{\cancel{s_n}})}{\sum^{N} [g_1(x_n)]^2} = \underline{\frac{1}{11.26} \frac{\sum^{N} y_n}{N}}.$$

6.

η in t iteration

$$\alpha_t = \eta = \frac{\sum g_t(x_n)(y_n - s_n^{t-1})}{\sum g_t^2(x_n)}$$

$$\Rightarrow \alpha_t \sum g_t^2(x_n) = \sum g_t(x_n) \cdot y_n - \sum g_t(x_n) s_n^{t-1}$$

$$\Rightarrow \sum g_t \left(\underbrace{\alpha_t \sum g_t + s_n^{n-1}}_{s_n^t}\right) = \sum g_t(x_n) \cdot y_n$$

$$\Rightarrow \sum^N s_n^t g_t(x_n) = \sum^N g_t(x_n) \cdot y_n$$

7. ~~By~~ Lecture 11. P19.

initial = 0.

~~$\alpha_1 = \eta^* = \frac{\sum^N g_t(x_n)(y_n - s_n^0)}{\sum^N g_t^2(x_n)}$~~

~~$= \frac{\sum^N g_t(x_n) \cdot y_n}{\sum^N g_t^2(x_n)}$~~

~~By Problem 6 $= \frac{\sum s_n^1 \cdot g_t(x_n)}{\sum^N g_t^2(x_n)} = \frac{\sum g_t^2(x_n)}{\sum g_t^2(x_n)} = 1$~~

~~$(s_n^1 = g_1(x_n))$~~

7.

Initially $s_1 = \dots = s_n = 0$

In the first iteration:

we find g_t by running squared error regression on $\{\{x_n, y_n\}\}$

after find g_t

when we find α_1

since we have to minimize $\min \frac{1}{N} \sum (y_n - \eta g_t(x_n))^2$

$\alpha_1 = \eta$ must be 1, since its the same objective function of regression in the first ~~itera~~ regression.

8. OR. $x_1 \dots x_d$ 多為 -1 才輸出 -1

otherwise $+1$

$(w_0, w_1, \dots, w_d)$

$= (d-1, 1, \dots, 1)$

when $x_i \ \forall i = -1$.

$\sum_i^d w_i x_i = -1 \Rightarrow$ ~~sign~~ $g_A(x) = -1$

9. For output layer.

$$\frac{\partial e_n}{\partial w_{i1}^{(L)}} = -2(y_n - s_1^{(L)}) \cdot (x_i^{(L-1)})$$

For other layer

$$\frac{\partial e_n}{w s_j^{(l)}} = \delta^{(l)} \cdot (x_1^{l-1})$$

By back prop: $\delta_j^{(l)} = \sum_i \delta_k^{(l+1)} (w_{jk}^{l+1}) \tanh'(s_j^{(l)})$

since $w_{ij}^{(l)} = 0 \Rightarrow \delta_j^{(l)} = 0 \quad \forall l < L, j$

only $x_0^{L-1} = 1$, $w_i^{L-1} = 0$

$\Rightarrow$ only $\frac{\partial e_n}{\partial w_{01}^{(L)}}$ may not be zero.

10.

$$\frac{\partial e}{\partial s_K^{(L)}} = \frac{\partial -\Sigma v_K \ln q_K}{\partial s_K}$$

$$= \frac{\partial -(v_1 \ln q_1 + v_2 \ln q_2 + \cdots + v_K \ln q_K + \cdots)}{\partial s_K}$$

$$= -\left(v_1 \frac{1}{q_1} \frac{\partial q_1}{\partial s_K} + \cdots + v_K \frac{1}{q_K} \frac{\partial q_K}{\partial s_K} \quad + v_K \frac{1}{q_K} \frac{\partial q_K}{\partial s_K}\right)$$

$$\begin{cases} \dfrac{\partial q_i}{\partial s_j} = -q_i q_j & \text{for } i \neq j \\ \dfrac{\partial q_i}{\partial q_i} = q_i(1-q_i] \end{cases}$$

$$= -\left(v_1 \frac{1}{\cancel{q_1}} \cancel{q_1} q_K + \cdots + v_K \frac{q_K}{q_K}(1-q_K) + \cdots\right)$$

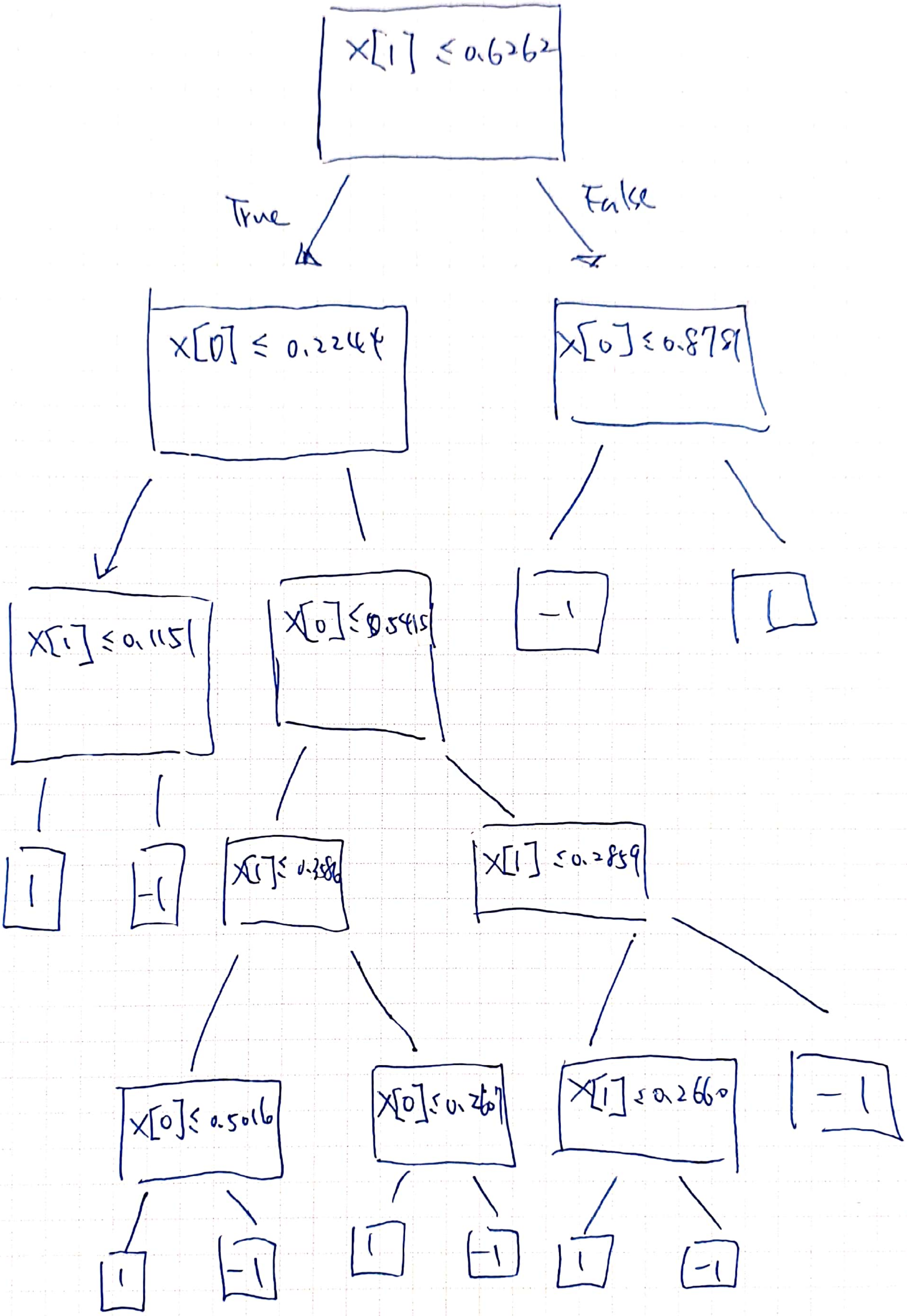
$$= -(-v_1 q_K - \cdots \quad - v_K q_K + v_K + \cdots)$$

since for $v_1 \cdots v_K \cdots v_K.$, only one will be 1

others are 0

$$= -(-q_K + v_K) = v_K - q_K. \#$$

11.



X[1] ≤ 0.6262
True
False
X[0] ≤ 0.2244
X[0] ≤ 0.8789
X[1] ≤ 0.1151
X[0] ≤ 0.5415
-1
1
1
-1
X[1] ≤ 0.3586
X[1] ≤ 0.2859
X[0] ≤ 0.5016
X[0] ≤ 0.2607
X[1] ≤ 0.2660
-1
1
-1
1
-1
1
-1

In [544]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [517]:

```
class Node:
    def __init__(self, b, col, value = None, height = 0, Mode = None):
        self.b = b
        self.col = col
        self.value = value
        self.LNode = None
        self.RNode = None
        self.height = height
        self.Mode = Mode
```

In [518]:

```
def Gini(y):
    Gini = 1
    try:
        N = y.shape[0]
    except:
        return Gini

    for k in [1, -1]:
        Gini += -(np.sum(y == k)/N)**2
    return Gini
```

In [519]:

```
def DStump(X, y):
    row, col = X.shape
    X_sort = np.sort(X, axis = 0)
    thresList = np.r_[[X_sort[0,:]-1], (X_sort[0:-1,:] + X_sort[1:,:])/2, [X_sort[-1
    minPurity = y.shape[0]
    DSb = 0
    DScol = 0
    for c in range(col):
        for i in range(thresList.shape[0]):

            y0 = y[ X[:,c] <  thresList[i,c]]
            y1 = y[ X[:,c] >= thresList[i,c]]
            Purity = y0.shape[0]*Gini(y0)+y1.shape[0]*Gini(y1)
            if minPurity > Purity:
                minPurity = Purity
                DSb = thresList[i,c]
                DScol = c

    return DSb, DScol
```

In [520]:

```
def DTreeFull(X, y):

    if (np.sum(y!=y[0])==0 or X.shape[0]==1 or np.sum(X!=X[0, :])==0):
        node = Node(None, None, y[0])
        return node

    DSb, DScol = DStump(X, y)
    #print(DSb,DScol)
    LX = X[ (X[:, DScol] < DSb) , :]
    RX = X[ (X[:, DScol] >= DSb), :]

    Ly = y[ (X[:, DScol] < DSb) ]
    Ry = y[ (X[:, DScol] >= DSb)]

    if (sum(y == -1) >sum(y == 1)):
        Mode = -1
    else:
        Mode = 1

    node = Node(DSb, DScol, Mode = Mode)


    node.LNode = DTree(LX, Ly)
    node.RNode = DTree(RX, Ry)


    return node
```

In [ ]:

In [521]:

```
data = pd.read_csv('hw3_train.dat', sep='\s+', header=None)
X_train = data.iloc[:,0:2].values
y_train = data.iloc[:,2].values
data = pd.read_csv('hw3_test.dat', sep='\s+', header=None)
X_test = data.iloc[:,0:2].values
y_test = data.iloc[:,2].values
```

In [ ]:

In [ ]:

In [522]:

```python
def nodeH(node):

    h = 0
    if node == None:

        return
    if node.LNode == None and node.RNode == None:

        if node.height > h:
            h = node.height
        return h

    if node.LNode != None:

        lh = nodeH(node.LNode)
    if node.RNode != None:

        rh = nodeH(node.RNode)

    print('MaxH=',h)
    return max(h,lh,rh)
```

In [523]:

```python
def internal_node(node):

    if node == None:

        return 0
    if node.LNode == None and node.RNode == None:
        print('==End',node.value, node.height)
        return 0
    l = 0; r = 0
    if node.LNode != None:

        print('L',node.b,node.col, node.height+1,'Mode=',node.Mode,'Value=',node.valu
        l = internal_node(node.LNode)
    if node.RNode != None:

        print('R',node.b,node.col,node.height+1,'Mode=',node.Mode,'Value=',node.valu
        r = internal_node(node.RNode)
    return 1 + l + r
```

In [524]:

```
def setNodeH(node):

    if node == None:

        return 0
    if node.LNode == None and node.RNode == None:

        return 0

    if node.LNode != None:
        LH = node.height + 1
        node.LNode.height = LH

        setNodeH(node.LNode)
    if node.RNode != None:
        RH = node.height + 1
        node.RNode.height = RH

        setNodeH(node.RNode)
    return node
```

In [525]:

```
node1 = node0
setNodeH(node0)
```

Out[525]:

```
<__main__.Node at 0x1a58e3d4908>
```

In [526]:

```
def NodePrune(node, MaxH = np.inf):

    if node == None:

        return
    if node.LNode == None and node.RNode == None:

        return

    if node.LNode != None:
        if node.LNode.height > MaxH:
            node.value = node.Mode
        node.LNode = NodePrune(node.LNode,MaxH-1)
    if node.RNode != None:
        if node.RNode.height > MaxH:
            node.value = node.Mode
        node.RNode = NodePrune(node.RNode,MaxH-1)

    return node
```

In [527]:

```
node0 = DTree(X_train, y_train)

nodeH(setNodeH(node0))
```

```
MaxH= 0
MaxH= 0
MaxH= 0
MaxH= 0
MaxH= 0
MaxH= 0
MaxH= 0
MaxH= 0
MaxH= 0
MaxH= 0

C:\Users\Morris\Anaconda3\lib\site-packages\ipykernel_launcher.py:9: RuntimeWarnin
g: invalid value encountered in long_scalars
  if __name__ == '__main__':
```

Out[527]:

```
5
```

In [528]:

```
nodeP = NodePrune(setNodeH(node0),2)
#internal_node(nodeP)
```

In [529]:

```
def predictDT(node, X,MaxH=np.inf):
    if node.value is not None:

        return node.value

    b = node.b
    c = node.col

    if node.height >= MaxH:
        #print('height' , node.height)
        return node.Mode

    if X[c] < b:
        return predictDT(node.LNode,X, MaxH)
    else:
        return predictDT(node.RNode,X, MaxH)

def predict(node, Xall,MaxH=np.inf):
    row = Xall.shape[0]
    ypred = np.zeros(row)
    for i in range(row):
        ypred[i] = predictDT(node,Xall[i,:],MaxH)
    return ypred

def err01(ypred, y):
    return np.sum(ypred != y)/y.shape[0]
```

In [540]:

```
print('Ein=',err01(predict(node0,X_train),y_train))
```

```
Ein= 0.0
```

In [541]:

```
print('Eout=',err01(predict(node0,X_test),y_test))
```

```
Eout= 0.126
```

In [543]:

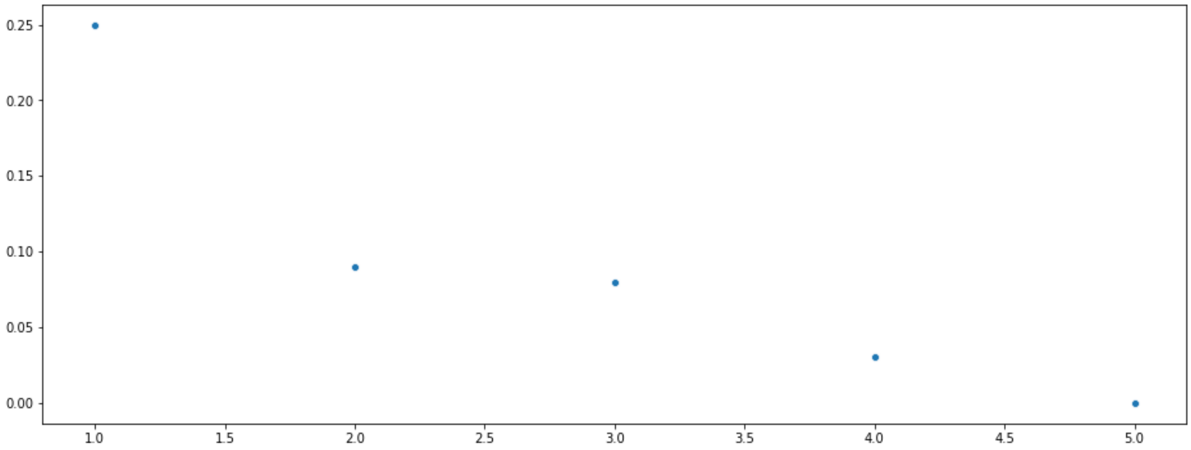
```
ein = []
eout = []
for h in range(5):

    ein = ein + [err01(predict(node0,X_train,h+1),y_train)]
    eout = eout + [err01(predict(node0,X_test,h+1),y_test)]
```

In [551]:

```
plt.figure(figsize=(16, 6))
sns.scatterplot(range(1,6),ein)
```

Out[551]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a58f1a27b8>
```



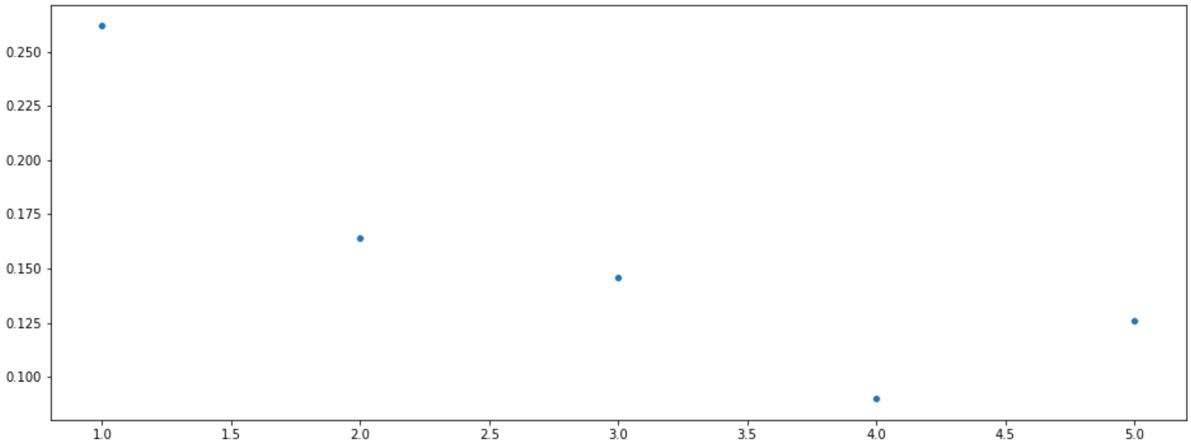
Q12: 大致上是高度越高，Ein越小。因為高度越高，可以branch越多次。

In [552]:

```
plt.figure(figsize=(16, 6))
sns.scatterplot(range(1,6),eout)
```

Out[552]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a58f1fbb00>
```



Q13:不一定是高度越高，Eout越小。反在在H=4時,Eout最低

In [145]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
class Node:
    def __init__(self, b, col, value = None, height = 0, Mode = None):
        self.b = b
        self.col = col
        self.value = value
        self.LNode = None
        self.RNode = None
        self.height = height
        self.Mode = Mode
```

In [3]:

```
def Gini(y):
    Gini = 1
    try:
        N = y.shape[0]
    except:
        return Gini

    for k in [1, -1]:
        Gini += -(np.sum(y == k)/N)**2
    return Gini
```

In [4]:

```
def DStump(X, y):
    row, col = X.shape
    X_sort = np.sort(X, axis = 0)
    thresList = np.r_[[X_sort[0,:]-1], (X_sort[0:-1,:] + X_sort[1:,:])/2, [X_sort[-1
    minPurity = y.shape[0]
    DSb = 0
    DScol = 0
    for c in range(col):
        for i in range(thresList.shape[0]):

            y0 = y[ X[:,c] <  thresList[i,c]]
            y1 = y[ X[:,c] >= thresList[i,c]]
            Purity = y0.shape[0]*Gini(y0)+y1.shape[0]*Gini(y1)
            if minPurity > Purity:
                minPurity = Purity
                DSb = thresList[i,c]
                DScol = c

    return DSb, DScol
```

In [13]:

```
def DTree(X, y):

    if (np.sum(y!=y[0])==0 or X.shape[0]==1 or np.sum(X!=X[0, :])==0):
        node = Node(None, None, y[0])
        return node

    DSb, DScol = DStump(X, y)
    #print(DSb,DScol)
    LX = X[ (X[:, DScol] < DSb) , :]
    RX = X[ (X[:, DScol] >= DSb), :]

    Ly = y[ (X[:, DScol] < DSb) ]
    Ry = y[ (X[:, DScol] >= DSb)]

    if (sum(y == -1) >sum(y == 1)):
        Mode = -1
    else:
        Mode = 1

    node = Node(DSb, DScol, Mode = Mode)


    node.LNode = DTree(LX, Ly)
    node.RNode = DTree(RX, Ry)


    return node
```

In [ ]:

In [6]:

```
data = pd.read_csv('hw3_train.dat', sep='\s+', header=None)
X_train = data.iloc[:,0:2].values
y_train = data.iloc[:,2].values
data = pd.read_csv('hw3_test.dat', sep='\s+', header=None)
X_test = data.iloc[:,0:2].values
y_test = data.iloc[:,2].values
```

In [ ]:

In [ ]:

In [7]:

```
def nodeH(node):

    h = 0
    if node == None:

        return
    if node.LNode == None and node.RNode == None:

        if node.height > h:
            h = node.height
        return h

    if node.LNode != None:

        lh = nodeH(node.LNode)
    if node.RNode != None:

        rh = nodeH(node.RNode)

    print('MaxH=',h)
    return max(h,lh,rh)
```

In [8]:

```
def internal_node(node):

    if node == None:

        return 0
    if node.LNode == None and node.RNode == None:
        print('==End',node.value, node.height)
        return 0
    l = 0; r = 0
    if node.LNode != None:

        print('L',node.b,node.col, node.height+1,'Mode=',node.Mode,'Value=',node.valu
        l = internal_node(node.LNode)
    if node.RNode != None:

        print('R',node.b,node.col,node.height+1,'Mode=',node.Mode,'Value=',node.valu
        r = internal_node(node.RNode)
    return 1 + l + r
```

In [9]:

```
def setNodeH(node):

    if node == None:

        return 0
    if node.LNode == None and node.RNode == None:

        return 0

    if node.LNode != None:
        LH = node.height + 1
        node.LNode.height = LH

        setNodeH(node.LNode)
    if node.RNode != None:
        RH = node.height + 1
        node.RNode.height = RH

        setNodeH(node.RNode)
    return node
```

In [43]:

```
def predictDT(node, X,MaxH=np.inf):
    if node.value is not None:

        return node.value

    b = node.b
    c = node.col

    if node.height >= MaxH:
        #print('height' , node.height)
        return node.Mode

    if X[c] < b:
        return predictDT(node.LNode,X, MaxH)
    else:
        return predictDT(node.RNode,X, MaxH)

def predict(node, Xall,MaxH=np.inf):
    row = Xall.shape[0]
    ypred = np.zeros(row)
    for i in range(row):
        ypred[i] = predictDT(node,Xall[i,:],MaxH)
    return ypred

def err01(ypred, y):
    return np.sum(ypred != y)/y.shape[0]
```

In [39]:

```
data = pd.read_csv('hw3_train.dat', sep='\s+', header=None)
X_train = data.iloc[:,0:2].values
y_train = data.iloc[:,2].values
```

In [139]:

```
data_test = pd.read_csv('hw3_test.dat', sep='\s+', header=None)
X_test = data_test.iloc[:,0:2].values
y_test = data_test.iloc[:,2].values
```

In [154]:

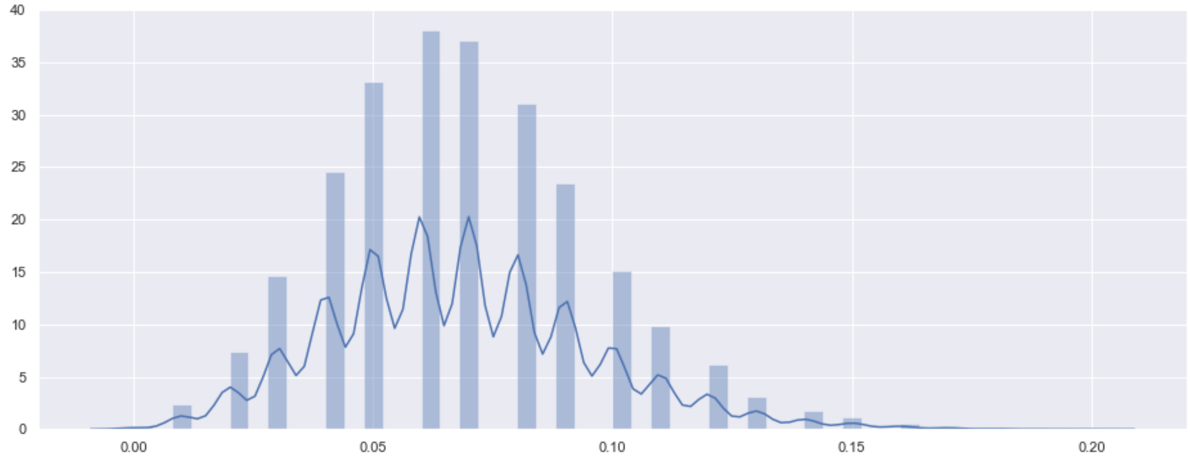
```
err_gt = []
err_rf_in = []
err_rf_out = []
RF_pred_in = np.zeros(100)
RF_pred_out = np.zeros(1000)
T = 30000
for i in range(T):
    data_bs = data.sample(n=80, replace=True)
    X_train_bs = data_bs.iloc[:,0:2].values
    y_train_bs = data_bs.iloc[:,2].values
    node_bs = setNodeH(DTreeFull(X_train_bs, y_train_bs))

    y_pred_in = predict(node_bs,X_train)
    RF_pred_in = RF_pred_in + y_pred_in/T

    y_pred_out = predict(node_bs,X_test)
    RF_pred_out = RF_pred_out + y_pred_out/T


    predict(node_bs,X_train)
    err_gt = err_gt + [err01(predict(node_bs,X_train),y_train)]
    err_rf_in = err_rf_in + [err01(np.sign(RF_pred_in), y_train)]
    err_rf_out = err_rf_out + [err01(np.sign(RF_pred_out), y_test)]
```

```
C:\Users\Morris\Anaconda3\lib\site-packages\ipykernel_launcher.py:9: RuntimeWarnin
g: invalid value encountered in long_scalars
  if __name__ == '__main__':
```

Q14:

In [163]:

```

sns.set()
plt.figure(figsize=(16, 6))
ax = sns.distplot(err_gt)
```



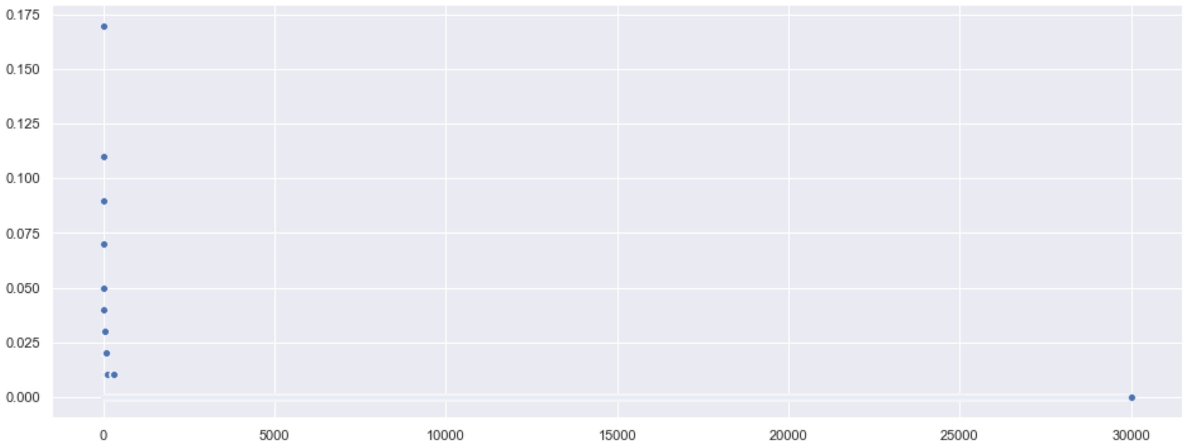
Q15: 第一張圖是Ein,第二是Eout。第三、四張是只畫前500個。/

In [156]:

```
plt.figure(figsize=(16, 6))
sns.scatterplot(range(T),err_rf_in)
```

Out[156]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x19445e81a90>
```

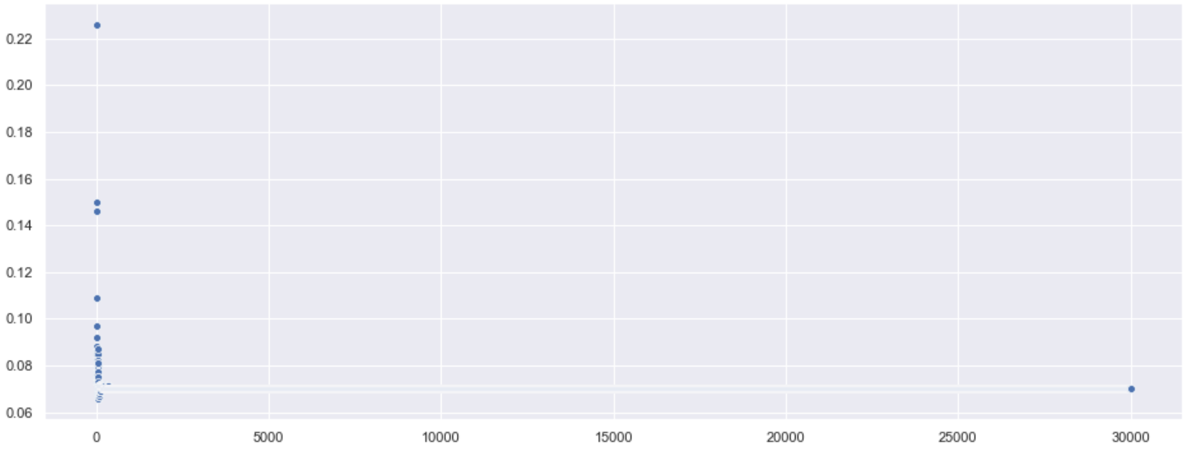


In [157]:

```
plt.figure(figsize=(16, 6))
sns.scatterplot(range(T),err_rf_out)
```

Out[157]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x194430aed30>
```

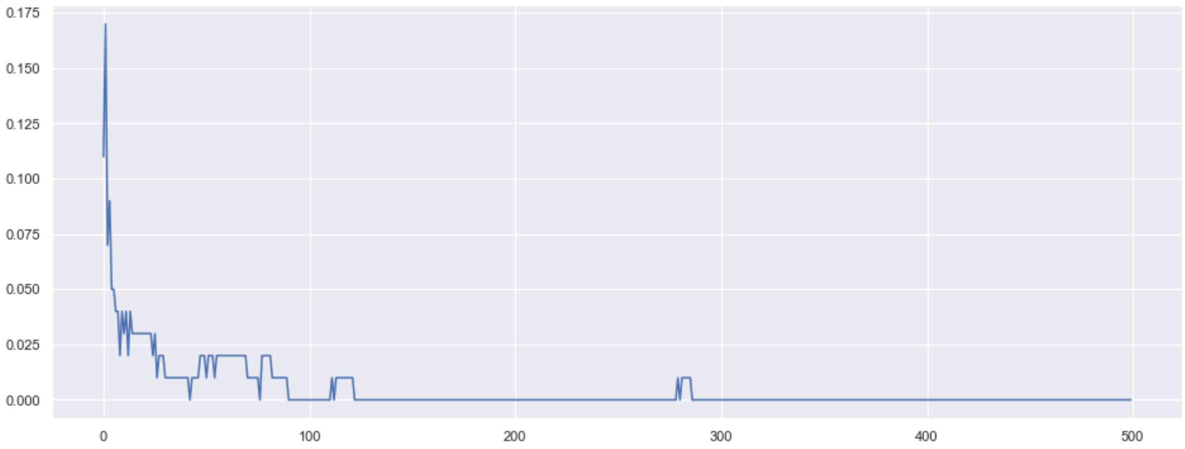


In [173]:

```
plt.figure(figsize=(16, 6))
sns.lineplot(range(500),err_rf_in[0:500])
```

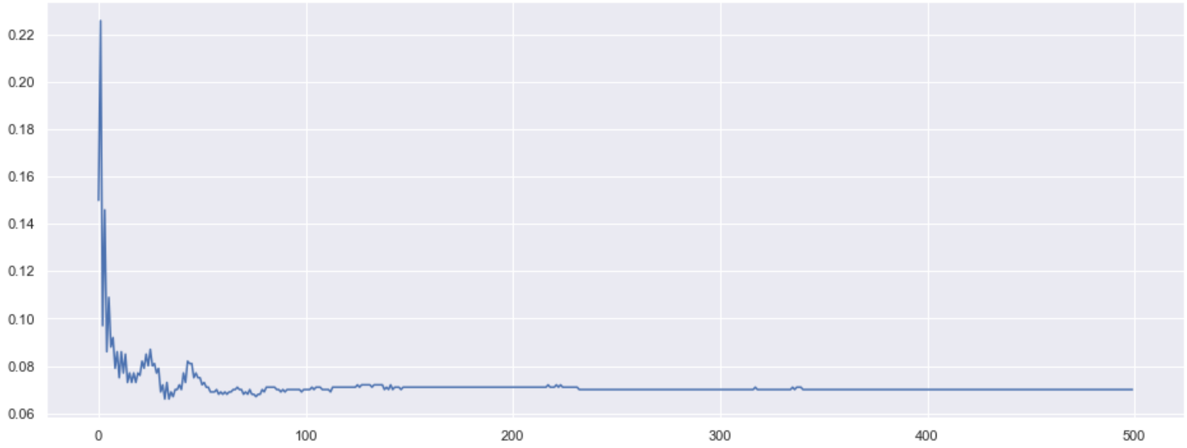
Out[173]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1944416cba8>
```



```
In [174]: plt.figure(figsize=(16, 6))
          sns.lineplot(range(500),err_rf_out[0:500])
```

```
Out[174]: <matplotlib.axes._subplots.AxesSubplot at 0x19445e1c860>
```



Q16:從最後兩張圖來看，Ein在前100次下降的非常快，在100次之後就幾乎為0。 但Eout不會完全跑到0，最後大概在0.05

```
In [ ]:
```