In [35]:

```
import pandas as pd
import graphviz
import numpy as np
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score
from sklearn import svm,preprocessing
from sklearn.model_selection import KFold, cross_val_score
from sklearn.neighbors import KNeighborsClassifier
```

In [36]:

```
data=pd.read_csv('CE802_Ass_2018_Data.csv')
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 15 columns):
F1       1500 non-null float64
F2       1500 non-null float64
F3       1500 non-null float64
F4       1500 non-null float64
F5       1500 non-null float64
F6       1500 non-null float64
F7       1500 non-null float64
F8       1500 non-null float64
F9       1500 non-null float64
F10      1500 non-null float64
F11      1500 non-null float64
F12      1500 non-null float64
F13      1500 non-null float64
F14      1500 non-null float64
Class    1500 non-null bool
dtypes: bool(1), float64(14)
memory usage: 165.6 KB
None
```

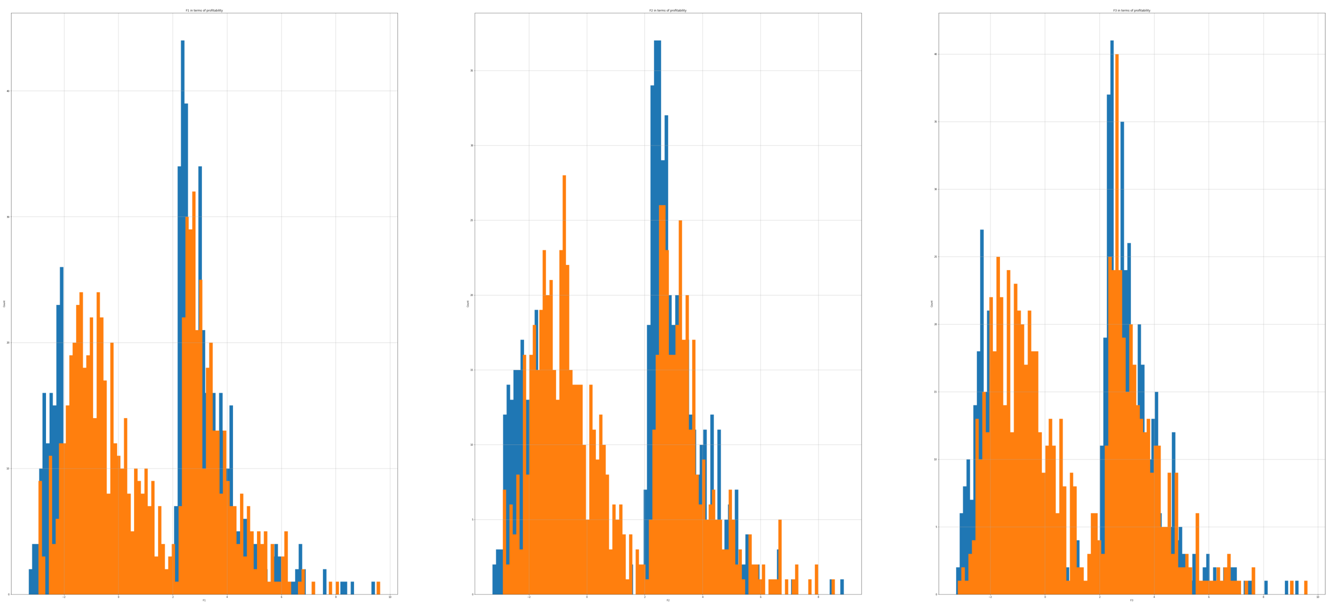
In [37]:

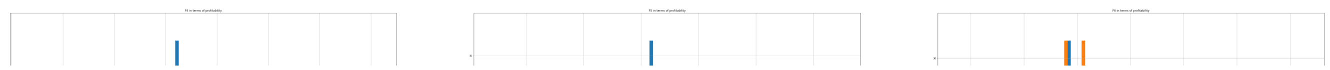
```
posf1=data[data['Class']==True]['F1']
negf1=data[data['Class']==False]['F1']
posf2=data[data['Class']==True]['F2']
negf2=data[data['Class']==False]['F2']
posf3=data[data['Class']==True]['F3']
negf3=data[data['Class']==False]['F3']
posf4=data[data['Class']==True]['F4']
negf4=data[data['Class']==False]['F4']
posf5=data[data['Class']==True]['F5']
negf5=data[data['Class']==False]['F5']
posf6=data[data['Class']==True]['F6']
negf6=data[data['Class']==False]['F6']
posf7=data[data['Class']==True]['F7']
negf7=data[data['Class']==False]['F7']
posf8=data[data['Class']==True]['F8']
negf8=data[data['Class']==False]['F8']
posf9=data[data['Class']==True]['F9']
negf9=data[data['Class']==False]['F9']
posf10=data[data['Class']==True]['F10']
negf10=data[data['Class']==False]['F10']
posf11=data[data['Class']==True]['F11']
negf11=data[data['Class']==False]['F11']
posf12=data[data['Class']==True]['F12']
negf12=data[data['Class']==False]['F12']
posf13=data[data['Class']==True]['F13']
negf13=data[data['Class']==False]['F13']
posf14=data[data['Class']==True]['F14']
```

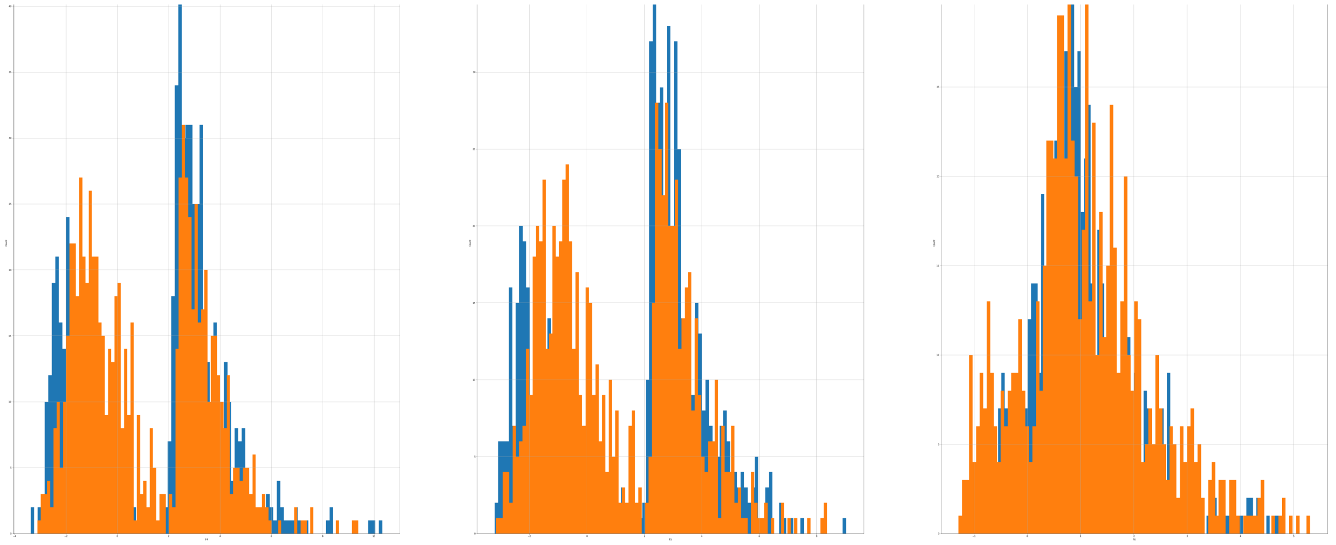
```
negf14=data[data['Class']==False]['F14']
```

In [38]:

```
fig=plt.figure(figsize=(100,100))
ax=fig.add_subplot(2,3,1)
ax.set_xlabel('F1')
ax.set_ylabel('Count')
plt.title('F1 in terms of profitability')
posf1.hist(bins=100,label='Positive')
negf1.hist(bins=100,label='Negative')

ax2=fig.add_subplot(2,3,2)
ax2.set_xlabel('F2')
ax2.set_ylabel('Count')
plt.title('F2 in terms of profitability')
posf2.hist(bins=100,label='Positive')
negf2.hist(bins=100,label='Negative')

ax3=fig.add_subplot(2,3,3)
ax3.set_xlabel('F3')
ax3.set_ylabel('Count')
plt.title('F3 in terms of profitability')
posf3.hist(bins=100,label='Positive')
negf3.hist(bins=100,label='Negative')

ax4=fig.add_subplot(2,3,4)
ax4.set_xlabel('F4')
ax4.set_ylabel('Count')
plt.title('F4 in terms of profitability')
posf4.hist(bins=100,label='Positive')
negf4.hist(bins=100,label='Negative')

ax5=fig.add_subplot(2,3,5)
ax5.set_xlabel('F5')
ax5.set_ylabel('Count')
plt.title('F5 in terms of profitability')
posf5.hist(bins=100,label='Positive')
negf5.hist(bins=100,label='Negative')

ax6=fig.add_subplot(2,3,6)
ax6.set_xlabel('F6')
ax6.set_ylabel('Count')
plt.title('F6 in terms of profitability')
posf6.hist(bins=100,label='Positive')
negf6.hist(bins=100,label='Negative')
plt.show()
```

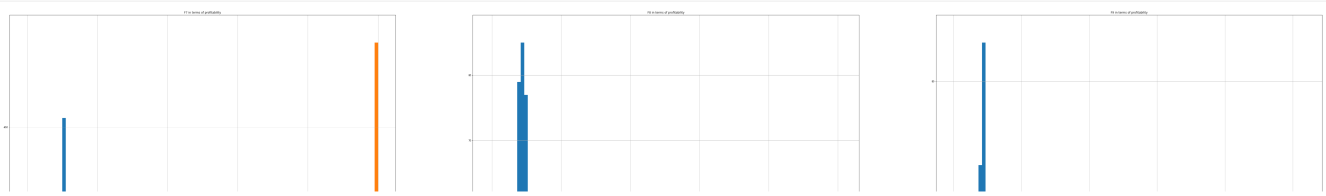


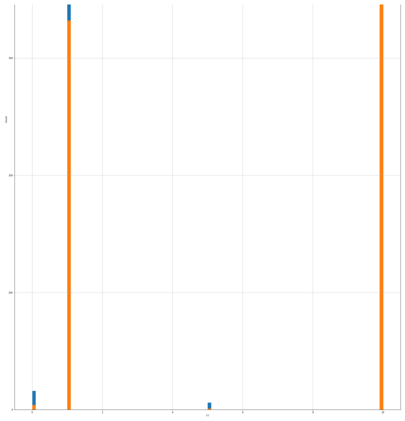
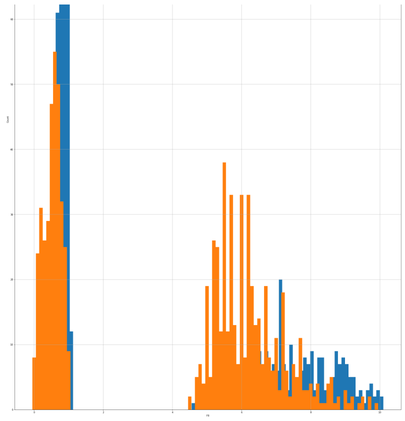
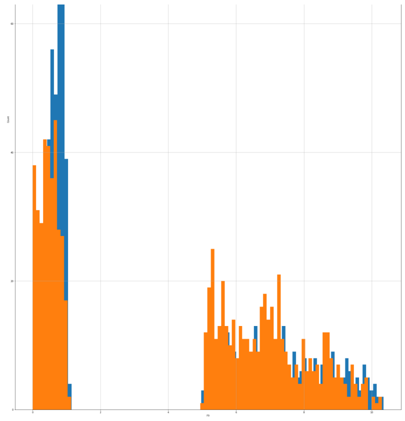


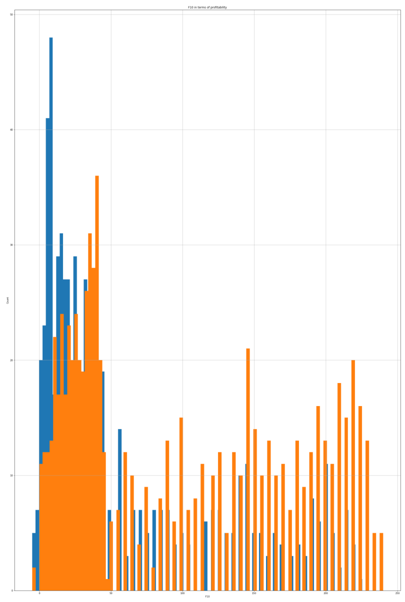
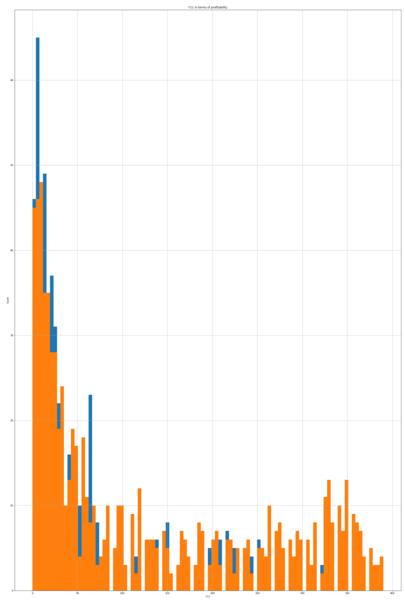
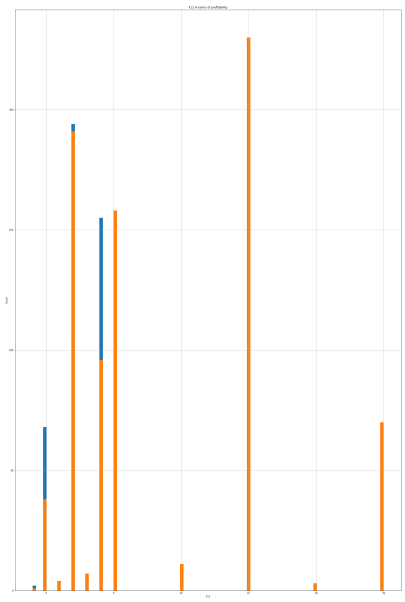


In [39]:

```
fig2=plt.figure(figsize=(100,100))

ax7=fig2.add_subplot(2,3,1)
ax7.set_xlabel('F7')
ax7.set_ylabel('Count')
plt.title('F7 in terms of profitability')
posf7.hist(bins=100,label='Positive')
negf7.hist(bins=100,label='Negative')

ax8=fig2.add_subplot(2,3,2)
ax8.set_xlabel('F8')
ax8.set_ylabel('Count')
plt.title('F8 in terms of profitability')
posf8.hist(bins=100,label='Positive')
negf8.hist(bins=100,label='Negative')

ax9=fig2.add_subplot(2,3,3)
ax9.set_xlabel('F9')
ax9.set_ylabel('Count')
plt.title('F9 in terms of profitability')
posf9.hist(bins=100,label='Positive')
negf9.hist(bins=100,label='Negative')

ax10=fig2.add_subplot(2,3,4)
ax10.set_xlabel('F10')
ax10.set_ylabel('Count')
plt.title('F10 in terms of profitability')
posf10.hist(bins=100,label='Positive')
negf10.hist(bins=100,label='Negative')

ax11=fig2.add_subplot(2,3,5)
ax11.set_xlabel('F11')
ax11.set_ylabel('Count')
plt.title('F11 in terms of profitability')
posf11.hist(bins=100,label='Positive')
negf11.hist(bins=100,label='Negative')

ax12=fig2.add_subplot(2,3,6)
ax12.set_xlabel('F12')
ax12.set_ylabel('Count')
plt.title('F12 in terms of profitability')
posf12.hist(bins=100,label='Positive')
negf12.hist(bins=100,label='Negative')

plt.show()
```



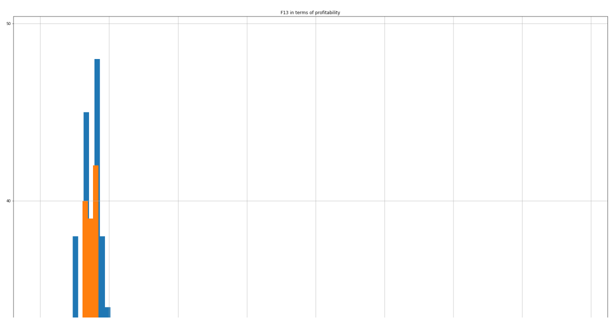
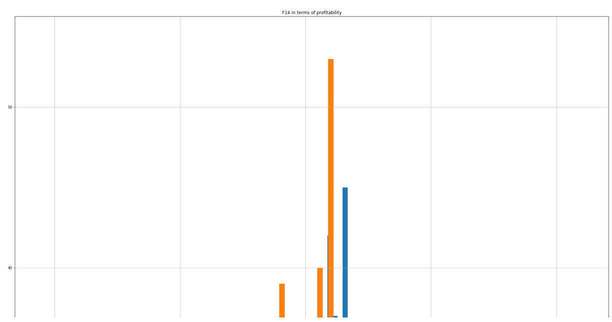




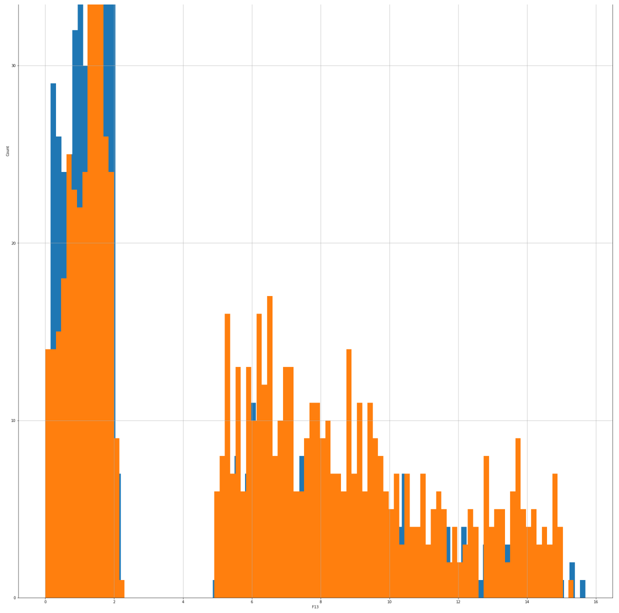
In [40]:

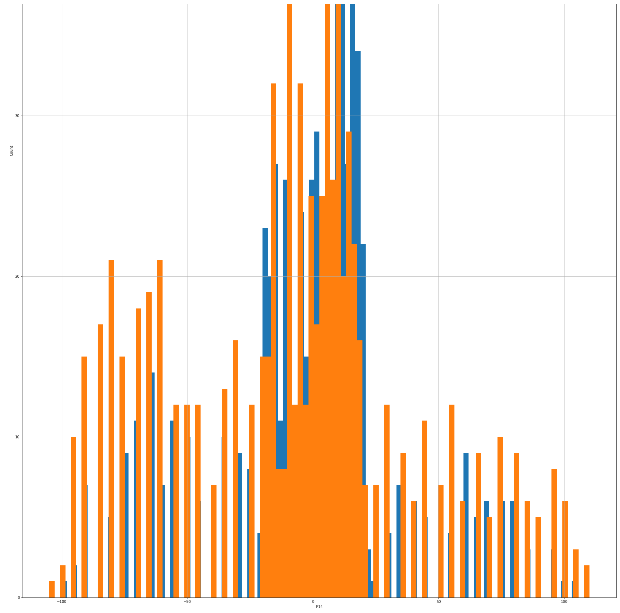
```
fig3=plt.figure(figsize=(100,100))

ax13=fig3.add_subplot(2,3,1)
ax13.set_xlabel('F13')
ax13.set_ylabel('Count')
plt.title('F13 in terms of profitability')
posf13.hist(bins=100,label='Positive')
negf13.hist(bins=100,label='Negative')

ax14=fig3.add_subplot(2,3,2)
ax14.set_xlabel('F14')
ax14.set_ylabel('Count')
plt.title('F14 in terms of profitability')
posf14.hist(bins=100,label='Positive')
negf14.hist(bins=100,label='Negative')
```

Out[40]:

<matplotlib.axes._subplots.AxesSubplot at 0x18776080>





In [41]:

```
features=['F1','F2','F3','F4','F5','F5','F6','F7','F8','F9','F10','F11','F12','F13','F14']
x=data[features]
minmaxscaler=preprocessing.MinMaxScaler(feature_range=(0,1))
x=minmaxscaler.fit_transform(x)
print(x)
y=data['Class']
clf=DecisionTreeClassifier(min_samples_split=100)
dt=clf.fit(x,y)

ddt=tree.export_graphviz(clf,out_file='tree.dot',feature_names=features,class_names='Class',filled=
True)
graph=graphviz.Source(ddt)

scores = cross_val_score(clf, x, y, cv=5)
print("Accuracy of a Decision Tree is:",round(np.mean((scores*100)),2))
```

```
[[0.55529776 0.54200988 0.47552448 ... 0.19230769 0.1044586  0.49767442]
 [0.04485692 0.33937397 0.1996892  ... 0.61538462 0.77388535 0.18604651]
 [0.56225831 0.44151565 0.43900544 ... 0.11538462 0.0866242  0.43255814]
 ...
 [0.46326373 0.54036244 0.45687646 ... 0.11538462 0.06496815 0.5627907 ]
 [0.65738592 0.82042834 0.65501166 ... 0.11538462 0.08535032 0.4372093 ]
 [0.43696829 0.44398682 0.5982906  ... 0.11538462 0.12101911 0.40930233]]
Accuracy of a Decision Tree is: 72.93
```

In [42]:

```
clf2 = svm.SVC(gamma=0.01, C=100.)
dtf=data[features]
minmaxscaler=preprocessing.MinMaxScaler(feature_range=(0,1))
dtf=minmaxscaler.fit_transform(dtf)
dtc=data['Class']
print(dtf,'\n\n')
print(dtc,'\n\n')
clf2=clf2.fit(dtf,dtc)
scores = cross_val_score(clf2, dtf, dtc, cv=5)
print("Accuracy of a Support Vector Machine is:",round(np.mean((scores*100)),2))
```

```
[[0.55529776 0.54200988 0.47552448 ... 0.19230769 0.1044586  0.49767442]
 [0.04485692 0.33937397 0.1996892  ... 0.61538462 0.77388535 0.18604651]
 [0.56225831 0.44151565 0.43900544 ... 0.11538462 0.0866242  0.43255814]
 ...
 [0.46326373 0.54036244 0.45687646 ... 0.11538462 0.06496815 0.5627907 ]
 [0.65738592 0.82042834 0.65501166 ... 0.11538462 0.08535032 0.4372093 ]
 [0.43696829 0.44398682 0.5982906  ... 0.11538462 0.12101911 0.40930233]]


0        False
1         True
```

```
1        True
2        True
3       False
4        True
5       False
6        True
7       False
8       False
9       False
10       True
11      False
12       True
13      False
14      False
15       True
16       True
17      False
18      False
19      False
20       True
21       True
22       True
23      False
24       True
25      False
26      False
27      False
28       True
29      False
        ...
1470     True
1471    False
1472     True
1473    False
1474     True
1475    False
1476     True
1477    False
1478    False
1479     True
1480     True
1481    False
1482    False
1483     True
1484    False
1485    False
1486    False
1487     True
1488     True
1489    False
1490     True
1491    False
1492    False
1493    False
1494    False
1495    False
1496    False
1497    False
1498    False
1499     True
Name: Class, Length: 1500, dtype: bool


Accuracy of a Support Vector Machine is: 74.73
```

In [43]:

```
clf3=KNeighborsClassifier(n_neighbors=3)
clf3.fit(dtf,dtc)
scores = cross_val_score(clf3, dtf, dtc, cv=5)
print("Accuracy of a 3NN Instance based Learning is:",round(np.mean((scores*100)),2))

data2=pd.read_csv('CE802_Ass_2018_Test.csv')
x1=data2[features]
minmaxscaler=preprocessing.MinMaxScaler(feature_range=(0,1))
x1=minmaxscaler.fit_transform(x1)
```

```
y1=data2['Class']
op1=clf2.predict(x1)
print(op1)
x1=data2[features]
df=pd.DataFrame(op1)
df.to_csv('Try.csv')
```

```
Accuracy of a 3NN Instance based Learning is: 72.33
[False  True False ... False False  True]
```

```
C:\Users\admin\Anaconda3\lib\site-packages\sklearn\preprocessing\data.py:323:
DataConversionWarning: Data with input dtype int64, float64 were all converted to float64 by
MinMaxScaler.
  return self.partial_fit(X, y)
```

In [ ]: