در این پروژه که مبتنی است بر انجام خوشه بندی بر اساس یک الگوریتم ensemble clustering شامل الگوریتم‌های grid-based و density-based، لازم است مدل بدست آمده را روی یک داده استریم ارزیابی کنیم. به این صورت که ابتدا داده ها را به صورت تصادفی، جایگشت داده، سپس آن را به یک سری window با اندازه های برابر تقسیم میکنیم. در هر مرحله، هر window را به داده های تست و آموزش تقسیم میکنیم و الگوریتم را روی آن اجرا میکنیم.

روند ساخت پروژه

کتابخانه ها:

در این پروژه از کتابخانه های زیر برای انجام پروژه بهره بردیم:

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

from sklearn.preprocessing import LabelEncoder
from sklearn.cluster import Birch
from sklearn.cluster import MiniBatchKMeans
from sklearn.cluster import SpectralClustering
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import KMeans

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
```

کتابخانه های پایه در بالا

کتابخانه های مربوط به پیش پردازش و انجام عمل خوشه بندی در وسط

و کتابخانه های مربوط به ارزیابی مدل ها را در پایین تصویر مشاهده می‌کنید

بارگذاری فایل و انجام پیش پردازش روی مجموعه داده:

مجموعه داده adults income به طور پیش فرض داده ها را به داده های آموزشی و تست تقسیم کرده است. در این پروژه، این دو مجموعه داده با هم ادغام شده و به یک مجموعه داده با 48842 رکورد تبدیل شده است. در مرحله ی پیش پردازش، ستون مربوط به اسم مقطع را به دلیل وجود کد آن مقطع تحصیلی حذف کردیم تا از بایاس شدن مدل جلوگیری شود. همچنین ستون های categorical با استفاده از تابع LabelEncoder به مقادیر عددی تبدیل شدند(داده های ناموجود هم یک دسته در نظر گرفته شدند). همچنین جایگشت دادن تصادفی داده ها در همین مرحله صورت گرفته است.

Loading the dataset and shuffling the dataset

```
df=pd.read_csv('adult_data.csv')
```

```
df=df.sample(frac=1)
df=df.dropna()
df
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
7212	31	Private	318647	11th	7	Never-married	Craft-repair	Not-in-family	White	Male	0	0	40	United-States	<=50K
17225	49	Private	239865	11th	7	Married-civ-spouse	Transport-moving	Husband	White	Male	0	0	40	United-States	<=50K
12827	53	Private	30244	Assoc-acdm	12	Married-civ-spouse	Exec-managerial	Husband	White	Male	15024	0	40	United-States	>50K
17980	36	Private	199947	Some-college	10	Divorced	Machine-op-inspct	Own-child	White	Female	0	0	30	United-States	<=50K
2381	17	Private	317681	11th	7	Never-married	Craft-repair	Own-child	White	Male	0	0	10	United-States	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
38713	19	?	264228	Some-college	10	Never-married	?	Own-child	White	Female	0	0	12	United-States	<=50K
4062	18	Private	263024	HS-grad	9	Never-married	Sales	Own-child	White	Female	0	0	40	United-States	<=50K
22083	38	Private	149018	HS-grad	9	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	40	United-States	<=50K
2593	39	Private	315776	Masters	14	Never-married	Exec-managerial	Not-in-family	Black	Male	8614	0	52	United-States	>50K
16206	30	Private	231413	Some-college	10	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	40	United-States	>50K

48842 rows × 15 columns

بارگذاری مجموعه داده و جایگشت دادن تصادفی آن

Converting the categorical attributes to numerical

```
lb_make = LabelEncoder()
dfd=df.copy()
dfd=dfd.drop(['education'],axis=1)
dfd['workclass'] = lb_make.fit_transform(df['workclass'])
dfd['marital-status'] = lb_make.fit_transform(df['marital-status'])
dfd['occupation'] = lb_make.fit_transform(df['occupation'])
dfd['relationship'] = lb_make.fit_transform(df['relationship'])
dfd['race'] = lb_make.fit_transform(df['race'])
dfd['sex'] = lb_make.fit_transform(df['sex'])
dfd['native-country'] = lb_make.fit_transform(df['native-country'])
dfd['income'] = lb_make.fit_transform(df['income'])
dfd
```

	age	workclass	fnlwgt	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
7212	31	4	318647	7	4	3	1	4	1	0	0	40	39	0
17225	49	4	239865	7	2	14	0	4	1	0	0	40	39	0
12827	53	4	30244	12	2	4	0	4	1	15024	0	40	39	1
17980	36	4	199947	10	0	7	3	4	0	0	0	30	39	0
2381	17	4	317681	7	4	3	3	4	1	0	0	10	39	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
38713	19	0	264228	10	4	0	3	4	0	0	0	12	39	0
4062	18	4	263024	9	4	12	3	4	0	0	0	40	39	0
22083	38	4	149018	9	2	3	0	4	1	0	0	40	39	0
2593	39	4	315776	14	4	4	1	2	1	8614	0	52	39	1
16206	30	4	231413	10	2	3	0	4	1	0	0	40	39	1

48842 rows × 14 columns

تبدیل صفات Categorical به Numerical با استفاده از تابع LabelEncoder

Windowing:

در شکل زیر اندازه window ها را برابر 1000 و گام های خواندن داده استریم جدید را برابر 200 در نظر میگیریم. یعنی به ازای ورود 200 داده ی جدید مجددا عملیات خوشه بندی جدید را انجام میدهیم و مدلمان را به روز میکنیم. همچنین در هر window، هفتاد درصد داده ها را به منظور آموزش و سی درصد دیگر را به منظور تست تقسیم بندی میکنیم:

Windowing

```
window_size=1000
step_size=200
windows=[]
for i in np.arange(0,int(len(dfd)/window_size)*window_size,step_size):
    train = pd.DataFrame(dfd.iloc[i:i+int(window_size*0.7)])
    test = pd.DataFrame(dfd.iloc[i+int(window_size*0.7):i+window_size])
    windows.append(('train:', train, 'test:', test))
windowsLen=len(windows)
print(windowsLen)
```

```
240
```

```
def testTrain(i):
    window=windows[i]
    train=pd.DataFrame(window[1])
    test=pd.DataFrame(window[3])
    return test,train
```

ارزیاب:

در این پروژه علاوه بر دقت و میانگین دقت(که معیارهای ارزیابی استفاده شده در مقاله هستند)، از precision ، recall و F1 score برای ارزیابی مدل بهره میبریم.

```
def evaluateModel(realTestResults, pred):
    conf=confusion_matrix(realTestResults, pred)
    print('Confusion Matrix: \n',conf)
    accuracy=accuracy_score(realTestResults, pred)
    print('Accuracy: %f' % accuracy)
    # precision tp / (tp + fp)
    precision = precision_score(realTestResults, pred)
    print('Precision: %f' % precision)
    # recall: tp / (tp + fn)
    recall = recall_score(realTestResults, pred)
    print('Recall: %f' % recall)
    # f1: 2 tp / (2 tp + fp + fn)
    f1 = f1_score(realTestResults, pred)
    print('F1 score: %f' % f1)
    return accuracy
```

تابع ارزیابی مدل

ساخت مدل و ارزیابی آن:

سری اول

در این سری از ساخت مدل، به ساخت سه مدل نامبرده در شکل پرداختیم و نتایج آنرا در کنار مدل خوشه بندی ensemble ساخته شده توسط آنها را مشاهده می‌کنید.

Clustering by 3 unique clustering algorithms VS their ensemblemethod together

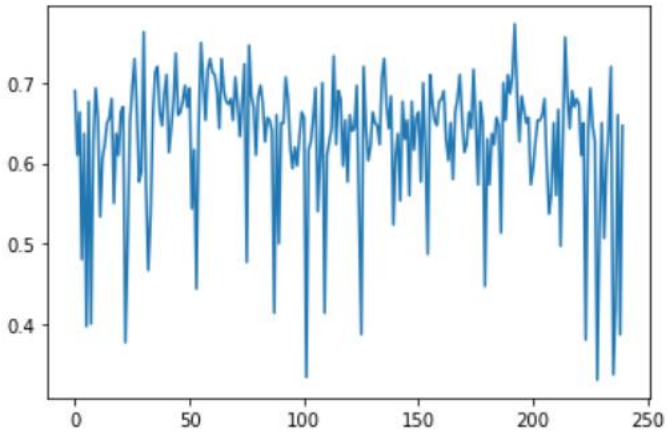
```
model1 = Birch(threshold=0.01, n_clusters=2)
model2 = MiniBatchKMeans(n_clusters=2)
model3 = SpectralClustering(n_clusters=2)
accuracies1=[]
accuracies2=[]
accuracies3=[]
accuracies4=[]
for i in range(windowsLen):
    test,train= testTrain(i)
    realTestResults=test['income']
    realTrainResults=train['income']
    test=test.drop(['income'],axis=1)
    train=train.drop(['income'],axis=1)
    model1.fit(train)
    pred1=model1.predict(test)
    model2.fit(train)
    pred2=model2.predict(test)
    model3.fit(train)
    pred3=model3.fit_predict(test)
    predComb=pd.concat([pd.DataFrame(pred1),pd.DataFrame(pred2),pd.DataFrame(pred3)],axis=1).mode(axis=1)[0]
    accuracies1.append(evaluateModel(realTestResults, pred1))
    print('\n')
    accuracies2.append(evaluateModel(realTestResults, pred2))
    print('\n')
    accuracies3.append(evaluateModel(realTestResults, pred3))
    print('\n')
    accuracies4.append(evaluateModel(realTestResults, predComb))
    print('\n ------------------------ \n')
```

ساخت و ارزیابی مدل ها

Birch

```
plt.plot(range(windowsLen),accuracies1)
print('average accuracy:',sum(accuracies1)/len(accuracies1))
```

```
average accuracy: 0.6311805555555556
```

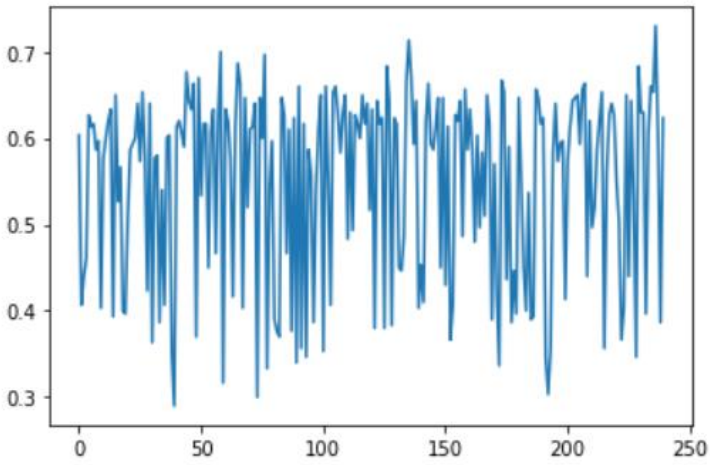


میانگین دقت و نحوه عملکرد مدل در طول زمان (دیدن window های جدید)

Mini Batch K-Means

```
plt.plot(range(windowsLen),accuracies2)
print('average accuracy:',sum(accuracies2)/len(accuracies2))
```

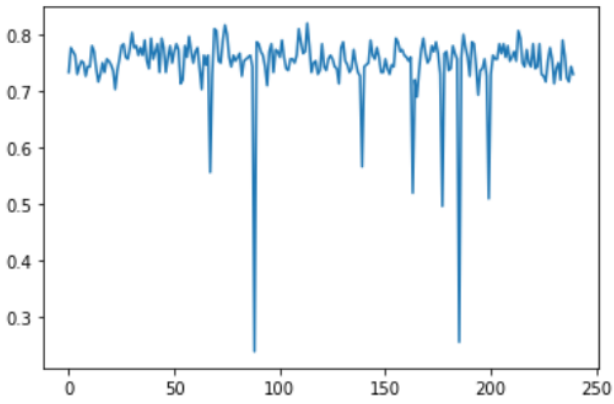
```
average accuracy: 0.5500555555555554
```



Spectral Clustering

```
plt.plot(range(windowsLen),accuracies3)
print('average accuracy:',sum(accuracies3)/len(accuracies3))
```

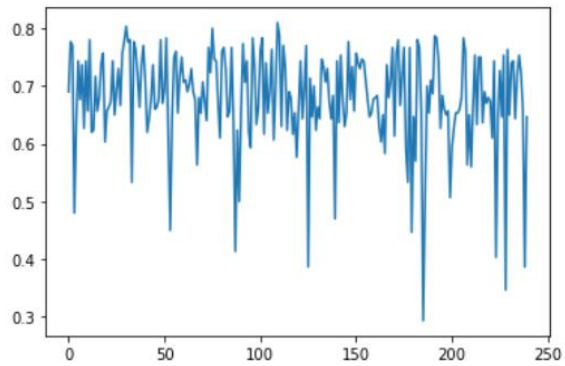
```
average accuracy: 0.7484861111111109
```



Ensemble method of the mentioned clustering algorithms

```
plt.plot(range(windowsLen),accuracies4)
print('average accuracy:',sum(accuracies4)/len(accuracies4))
```

```
average accuracy: 0.6791944444444445
```



ترکیب سه مدل قبلی به عنوان ensemble model

با وجود آنکه مدل ترکیبی ساخته شده میانگین دقت کمتری نسبت به روش Spectral Clustering دارد، اما در کل نتایج بهتری دارد. به این علت که روش Spectral Clustering به شدت نسبت به بایاس حساس است و به دلیل آنکه 75% مجموعه داده شامل کلاستر درآمد بیشتر از پنجاه هزار دلار است، مدل ساخته شده با استفاده از این روش در اکثر مواقع کلاستر را معادل درآمد بیشتر از پنجاه هزار دلار در نظر میگیرد که در واقع مدل ساخته شده با آن precision ، recall و F1 score پایینی نسبت به سایر روش ها دارد که در متن کد فرستاده شده نتایج آن را می‌توانید مشاهده و مقایسه کنید.

سری دوم

در این سری از ساخت مدل، هفت learner را به عنوان الگوریتم پایه به کار بردیم که جزئیات آن را به همراه نتایج در شکل های زیر مشاهده می‌کنید.

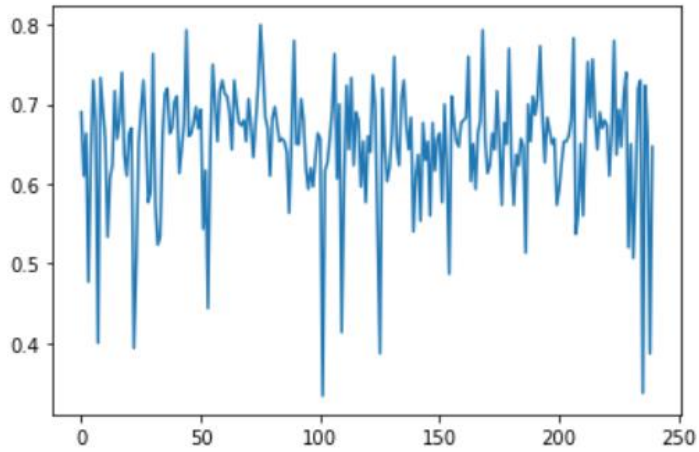
Ensemble Method 1: contains 3 Birch models with different thresholds, 1 Mini Batch K-Means model, 1 Spectral Clustering model, 1 Agglomerative Clustering model, 1 K-Means model as the base learners

```
model1 = Birch(threshold=0.01, n_clusters=2)
model2 = MiniBatchKMeans(n_clusters=2)
model3 = SpectralClustering(n_clusters=2)
model4 = Birch(threshold=0.1, n_clusters=2)
model5 = AgglomerativeClustering(n_clusters=2)
model6 = KMeans(n_clusters=2)
model7 = Birch(threshold=0.9, n_clusters=2)
accuracies=[]
for i in range(windowsLen):
    test,train= testTrain(i)
    realTestResults=test['income']
    realTrainResults=train['income']
    test=test.drop(['income'],axis=1)
    train=train.drop(['income'],axis=1)
    model1.fit(train)
    model2.fit(train)
    model3.fit(train)
    model4.fit(train)
    model5.fit(train)
    model6.fit(train)
    model7.fit(train)
    pred1=model1.predict(test)
    pred2=model2.predict(test)
    pred3=model3.fit_predict(test)
    pred4=model4.predict(test)
    pred5=model5.fit_predict(test)
    pred6=model6.fit_predict(test)
    pred7=model7.predict(test)
    pred1=pd.DataFrame(pred1)
    pred2=pd.DataFrame(pred2)
    pred3=pd.DataFrame(pred3)
    pred4=pd.DataFrame(pred4)
    pred5=pd.DataFrame(pred5)
    pred6=pd.DataFrame(pred6)
    pred7=pd.DataFrame(pred7)
    pred=pd.concat([pred1,pred2,pred3,pred4,pred5,pred6,pred7],axis=1).mode(axis=1)[0]
    accuracies.append(evaluateModel(realTestResults, pred))
    print('\n ---------------------- \n')
```

آموزش و تست مدل های به کار رفته و ارزیابی مدل ترکیبی (ensemble) آن ها

```
plt.plot(range(windowsLen),accuracies)
print('average accuracy:',sum(accuracies)/len(accuracies))
```

```
average accuracy: 0.6500277777777778
```



میانگین دقت و روند تغییرات دقت مدل در طول زمان

سری سوم

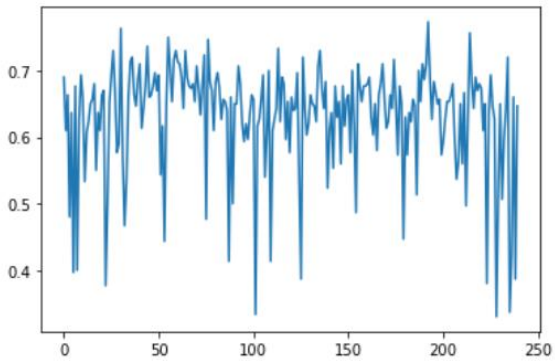
در سری سوم ساخت مدل، همگی learner های پایه الگوریتم Birch هستند با threshold های متفاوت. همانطور که در نتایج زیر و مقایسه آن با نتیجه مدل Birch موجود در سری اول میتوان دید، این دو نتایج بسیار نزدیکی به یکدیگر دارند و ساخت مدل ترکیبی از یک روش خوشه بندی، معمولا نتایج چندان بهتری به همراه نخواهد داشت.

Ensemble Method 2: contains only Birch models as base learners with different threshold

```
model1 = Birch(threshold=0.01, n_clusters=2)
model2 = Birch(threshold=0.2,n_clusters=2)
model3 = Birch(threshold=0.3,n_clusters=2)
model4 = Birch(threshold=0.4, n_clusters=2)
model5 = Birch(threshold=0.5,n_clusters=2)
model6 = Birch(threshold=0.7,n_clusters=2)
model7 = Birch(threshold=0.9, n_clusters=2)
model8 = Birch(threshold=1,n_clusters=2)
accuracies=[]
for i in range(windowsLen):
    test,train= testTrain(i)
    realTestResults=test['income']
    realTrainResults=train['income']
    test=test.drop(['income'],axis=1)
    train=train.drop(['income'],axis=1)
    model1.fit(train)
    model2.fit(train)
    model3.fit(train)
    model4.fit(train)
    model5.fit(train)
    model6.fit(train)
    model7.fit(train)
    model8.fit(train)
    pred1=model1.predict(test)
    pred2=model2.predict(test)
    pred3=model3.fit_predict(test)
    pred4=model4.predict(test)
    pred5=model5.fit_predict(test)
    pred6=model6.fit_predict(test)
    pred7=model7.predict(test)
    pred8=model8.predict(test)
    pred1=pd.DataFrame(pred1)
    pred2=pd.DataFrame(pred2)
    pred3=pd.DataFrame(pred3)
    pred4=pd.DataFrame(pred4)
    pred5=pd.DataFrame(pred5)
    pred6=pd.DataFrame(pred6)
    pred7=pd.DataFrame(pred7)
    pred8=pd.DataFrame(pred8)
    pred=pd.concat([pred1,pred2,pred3,pred4,pred5,pred6,pred7,pred8],axis=1).mode(axis=1)[0]
    accuracies.append(evaluateModel(realTestResults, pred))
    print('\n ------------------------ \n')
```

```
plt.plot(range(windowsLen),accuracies)
print('average accuracy:',sum(accuracies)/len(accuracies))
```

```
average accuracy: 0.6313194444444445
```



نتیجه گیری:

در تمامی سری های بالا، از روش های خوشه بندی ای بهره برده شد که density based یا grid based بودند. نتایج حاصل نشان میدهد که روش ترکیبی (ensemble) معمولا منجر به ساخت مدل بهتری می‌شود و همچنین از بایاس شدن مدل به خوشه ی اکثریت، جلوگیری میکند. اما مدل ترکیبی ساخته شده با استفاده از الگوریتم های به کار رفته، به مراتب دقت کمتری نسبت به دقت (میانگین دقت) ثبت شده در مقاله کسب کرده اند.