AMLHW4_p1p3p4

April 4, 2018

1 HW4

Chi Zhang UNI: cz2481
Jingyu Ren UNI: jr3738

```
In [49]: import numpy as np
         from scipy.io import loadmat
         import matplotlib.pyplot as plt
         from datetime import datetime, date, time
         from sklearn.covariance import EllipticEnvelope
         from sklearn.svm import OneClassSVM
         from sklearn.preprocessing import StandardScaler
         from sklearn.ensemble import IsolationForest
         from sklearn.manifold import TSNE
         from sklearn.decomposition import PCA
         from sklearn.metrics import roc_auc_score
         from sklearn.metrics import average_precision_score
         from sklearn.linear_model import LogisticRegression
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.model_selection import train_test_split
         from sklearn.model_selection import GridSearchCV
         from sklearn.model_selection import cross_validate
         from sklearn.pipeline import make_pipeline
         from imblearn.pipeline import make_pipeline as make_imb_pipeline
         from imblearn.over_sampling import RandomOverSampler
         from imblearn.under_sampling import RandomUnderSampler
         from sklearn.metrics import normalized_mutual_info_score
         from sklearn.metrics import adjusted_rand_score
         from sklearn.metrics import silhouette_samples, silhouette_score
         from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
```

```
In [6]: HW4=loadmat("/Users/albertzhang/Desktop/18spring/AML/HW/HW_4/annthyroid.mat")
```

```
In [46]: HW4.keys()
```

```
Out[46]: dict_keys(['y', '__version__', '__header__', 'X', '__globals__'])
```

```
In [47]: data=pd.DataFrame(HW4['X'])
         target=pd.DataFrame(HW4['y'])
```

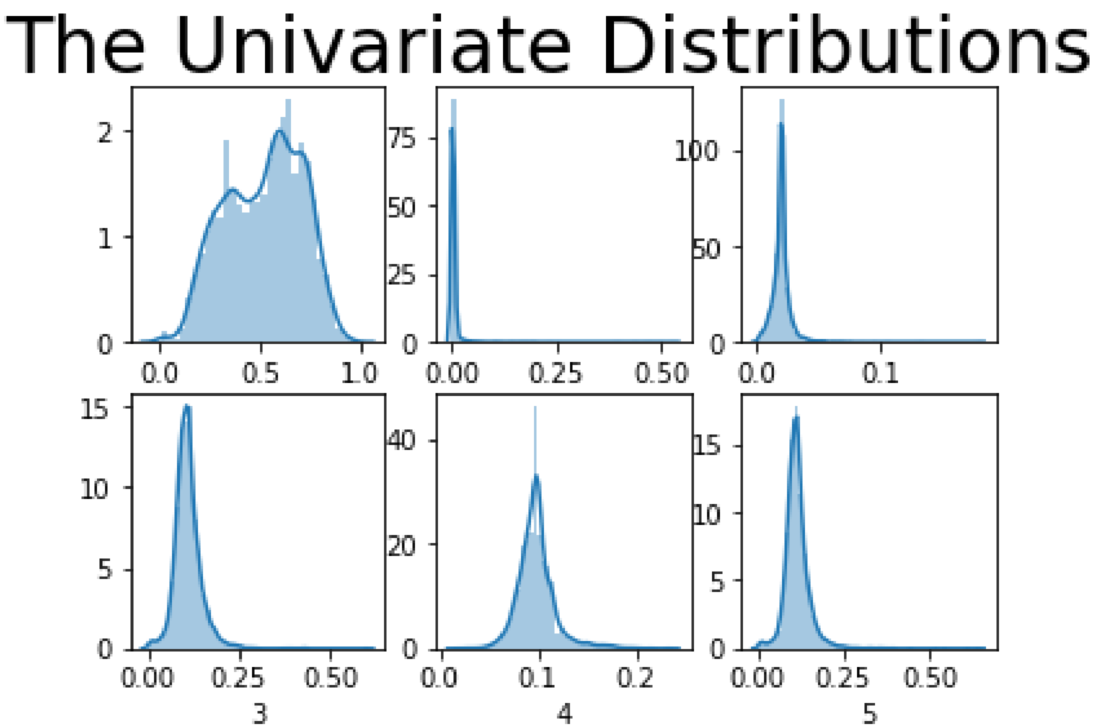
```
In [ ]: Y=HW4['y']
        Y=Y.tolist()
        y=[]
        for l in Y:
            y.append(l[0])
        target=y
```

2 Problem 1

2.1 1.1

2.1.1 Visualize the univariate distributions of all features

```
In [50]: import seaborn as sns

         plt.subplot(2, 3, 1)
         sns.distplot(data.iloc[:,0])
         plt.subplot(2, 3, 2)
         sns.distplot(data.iloc[:,1])
         plt.subplot(2, 3, 3)
         sns.distplot(data.iloc[:,2])
         plt.subplot(2, 3, 4)
         sns.distplot(data.iloc[:,3])
         plt.subplot(2, 3, 5)
         sns.distplot(data.iloc[:,4])
         plt.subplot(2, 3, 6)
         sns.distplot(data.iloc[:,5])
         plt.suptitle("The Univariate Distributions", fontsize=30)
         plt.show()
```

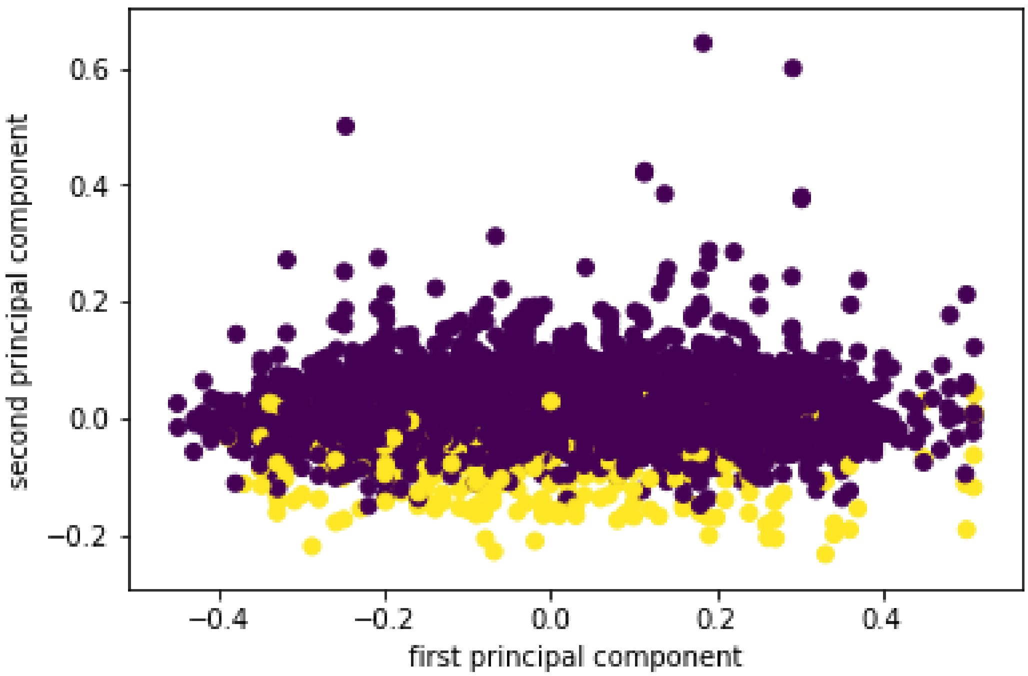


PCA for Visualization

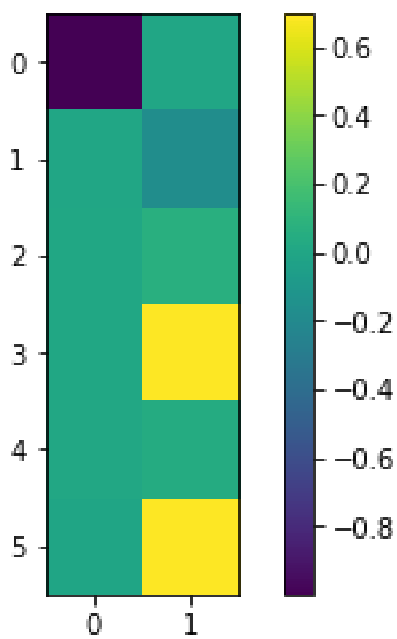
```
In [67]: from sklearn.decomposition import PCA
         print(data.shape)
         pca = PCA(n_components=2)
         X_pca = pca.fit_transform(data)
         print(X_pca.shape)
         plt.scatter(X_pca[:, 0], X_pca[:, 1], c=target)
         plt.xlabel("first principal component")
         plt.ylabel("second principal component")
```

```
(7200, 6)
(7200, 2)
```

```
Out[67]: Text(0,0.5,'second principal component')
```

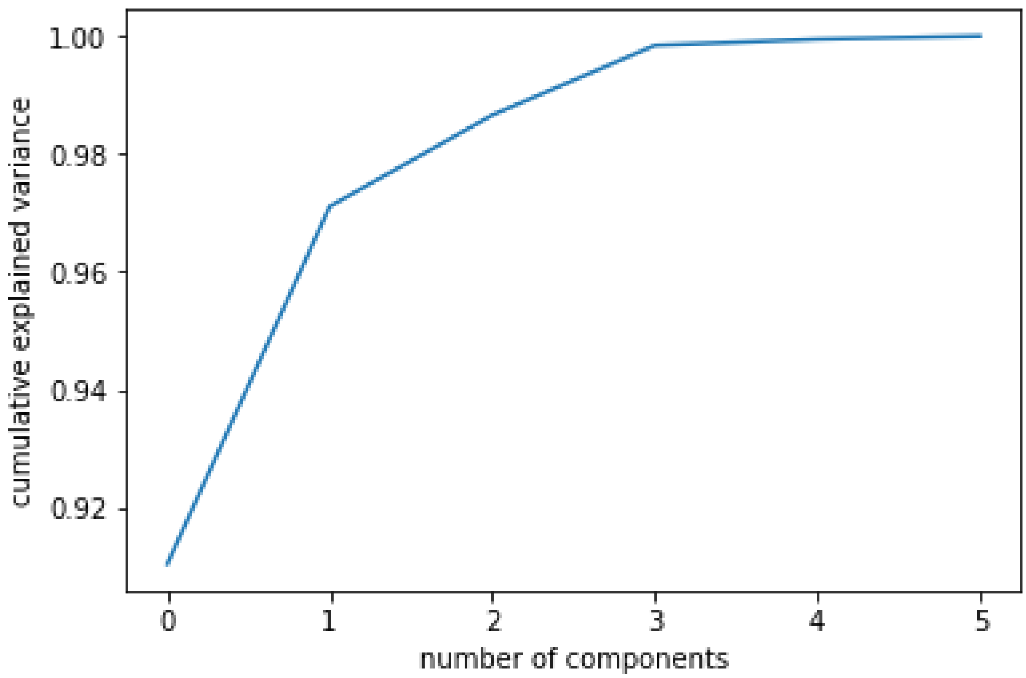



```
In [69]: components = pca.components_
         plt.imshow(components.T)
         plt.colorbar()
         plt.show()
```



In [70]:
```
pca_var = PCA().fit(data)
plt.plot(np.cumsum(pca_var.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
```

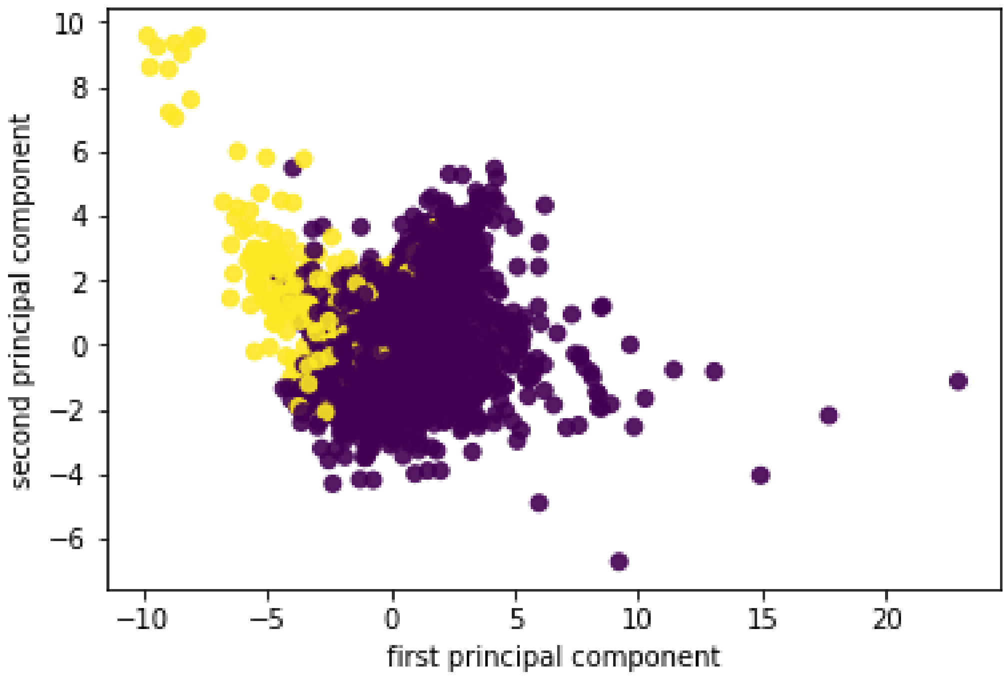
Out[70]: Text(0,0.5,'cumulative explained variance')



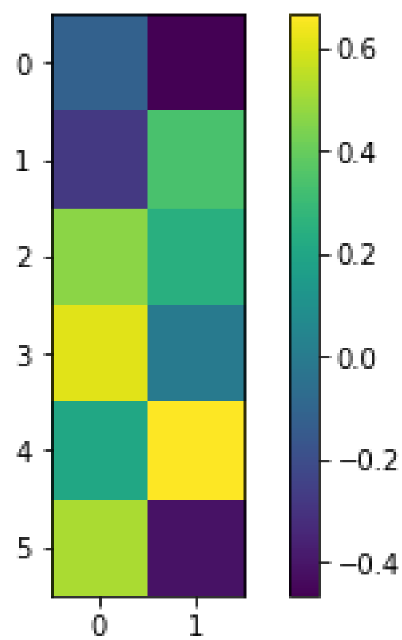
Answer: From the variance ratio graph, 2 components retain 98% of the variance.

In [86]:
```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
pca_scaled_com = make_pipeline(StandardScaler(), PCA(n_components=2))
X_pca_scaled = pca_scaled_com.fit_transform(data)
plt.scatter(X_pca_scaled[:, 0], X_pca_scaled[:, 1], c=target, alpha=.9)
plt.xlabel("first principal component")
plt.ylabel("second principal component")
```

Out[86]: Text(0,0.5,'second principal component')

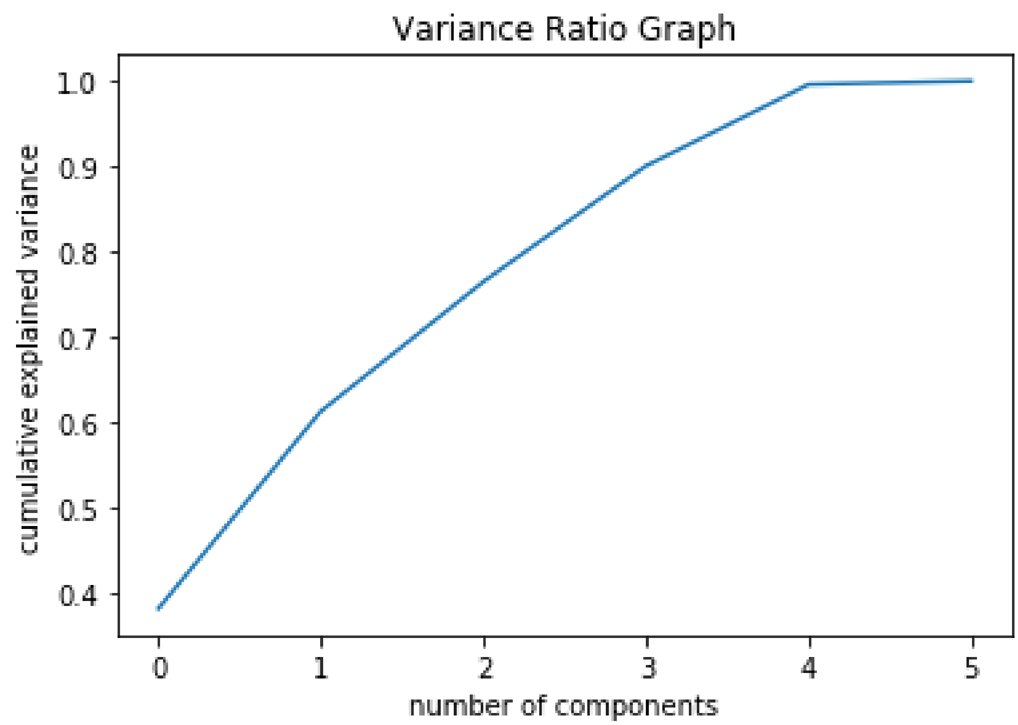



```
In [53]: components = pca_scaled.named_steps['pca'].components_
         plt.imshow(components.T)

         plt.colorbar()

Out[53]: <matplotlib.colorbar.Colorbar at 0x10b8267f0>
```



```
In [79]: from sklearn.preprocessing import StandardScaler
         scaler = StandardScaler().fit(data)
         data_scaled = scaler.transform(data)
         pca_var2 = PCA().fit(data_scaled)
         plt.plot(np.cumsum(pca_var2.explained_variance_ratio_))
         plt.xlabel('number of components')
         plt.ylabel('cumulative explained variance')
         plt.title('Variance Ratio Graph with scaling')
```

```
Out[79]: Text(0.5,1,'Variance Ratio Graph')
```



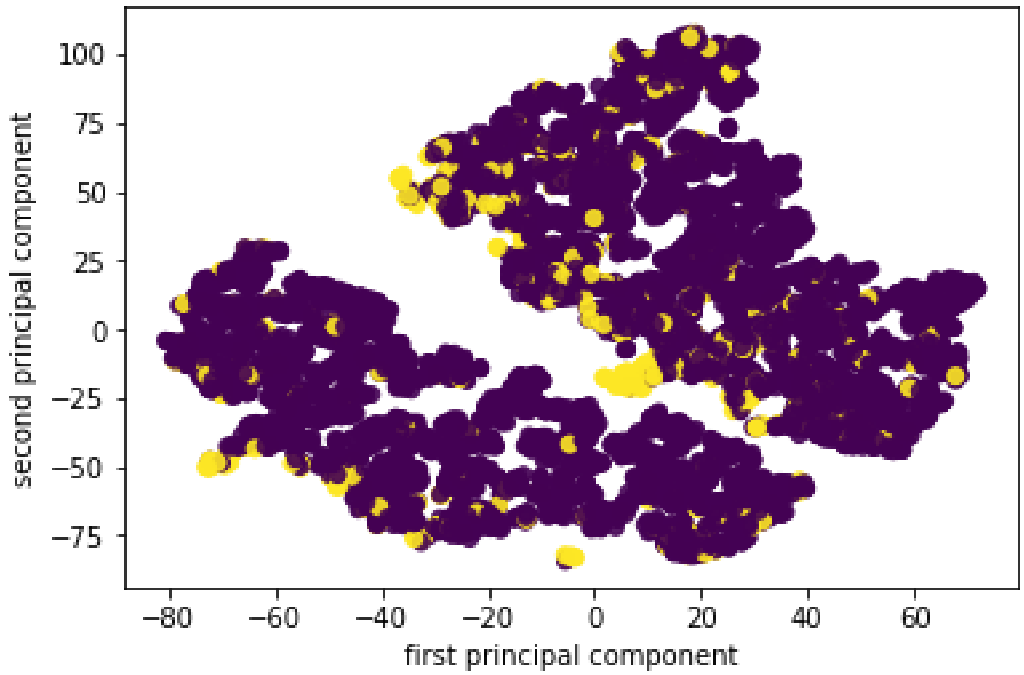
Answer: From the variance ratio with scaling graph, 3 components retain 90% of the variance.

2.2 1.2

Visualize the data using t-SNE

```
In [84]: from sklearn.manifold import TSNE
         X_tsne = TSNE(n_components=2).fit_transform(data)
         plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=target, alpha=.9)
         plt.xlabel("first principal component")
         plt.ylabel("second principal component")
```

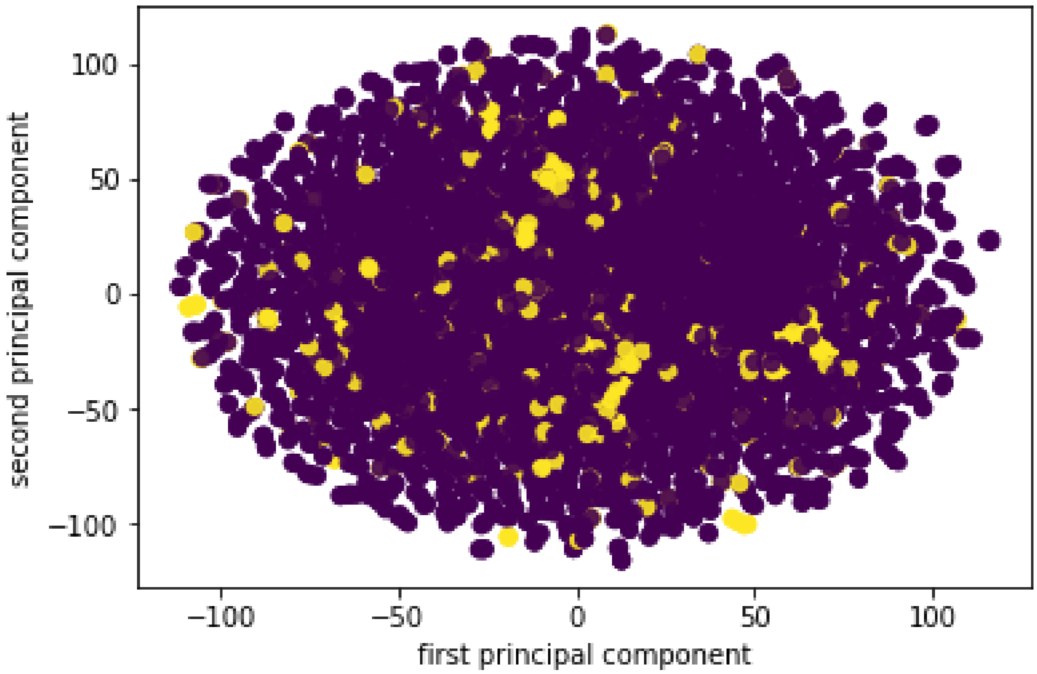
Out[84]: Text(0,0.5,'second principal component')



t-SNE with 2 components and tune the perplexity parameter equal to 2

In [85]:
```
X_tsne = TSNE(n_components=2,perplexity=2).fit_transform(data)
plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=target, alpha=.9)
plt.xlabel("first principal component")
plt.ylabel("second principal component")
```

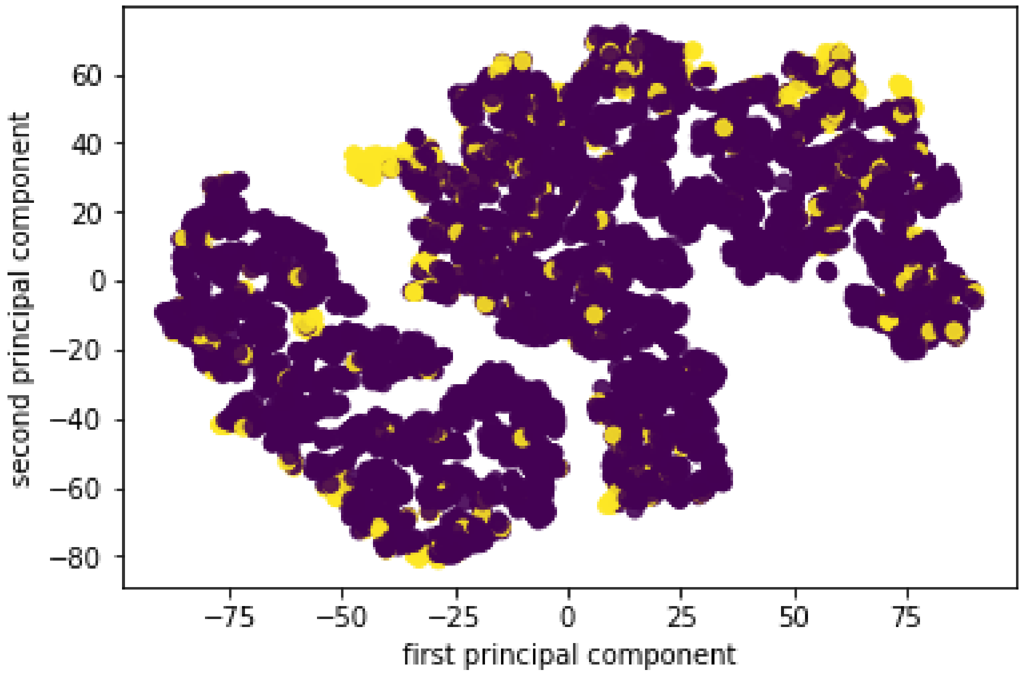
Out[85]: Text(0,0.5,'second principal component')



t-SNE with 2 components and tune the perplexity parameter equal to 30

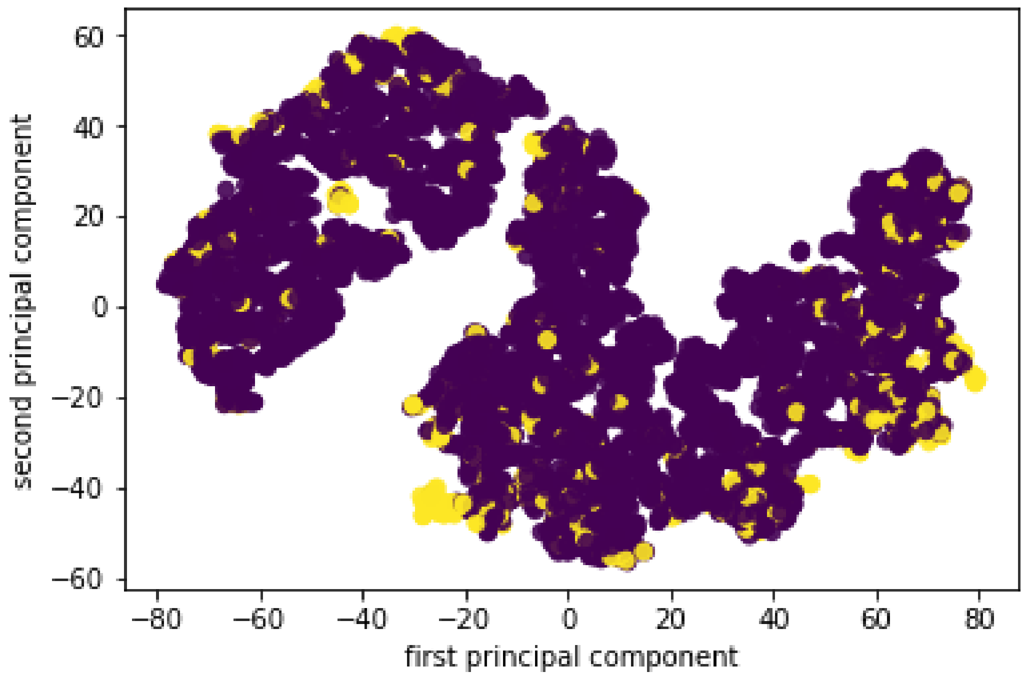
In [87]:
```
X_tsne = TSNE(n_components=2,perplexity=30).fit_transform(data)
plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=target, alpha=.9)
plt.xlabel("first principal component")
plt.ylabel("second principal component")
```

Out[87]:
```
Text(0,0.5,'second principal component')
```



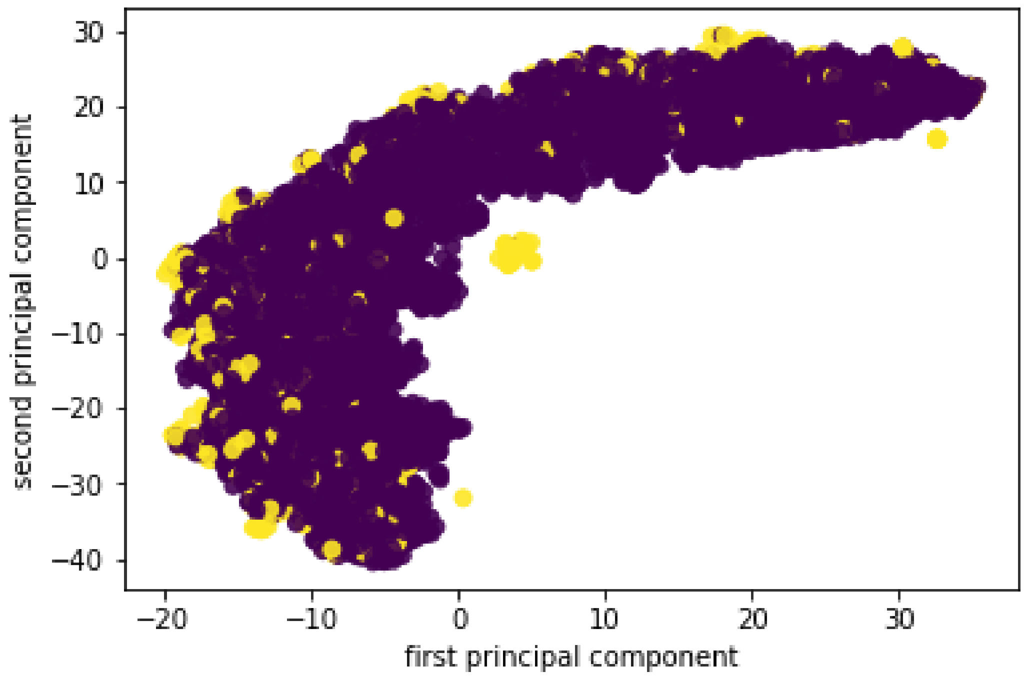
t-SNE with 2 components and tune the perplexity parameter equal to 50

```
In [88]: X_tsne = TSNE(n_components=2,perplexity=50).fit_transform(data)
         plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=target, alpha=.9)
         plt.xlabel("first principal component")
         plt.ylabel("second principal component")

Out[88]: Text(0,0.5,'second principal component')
```



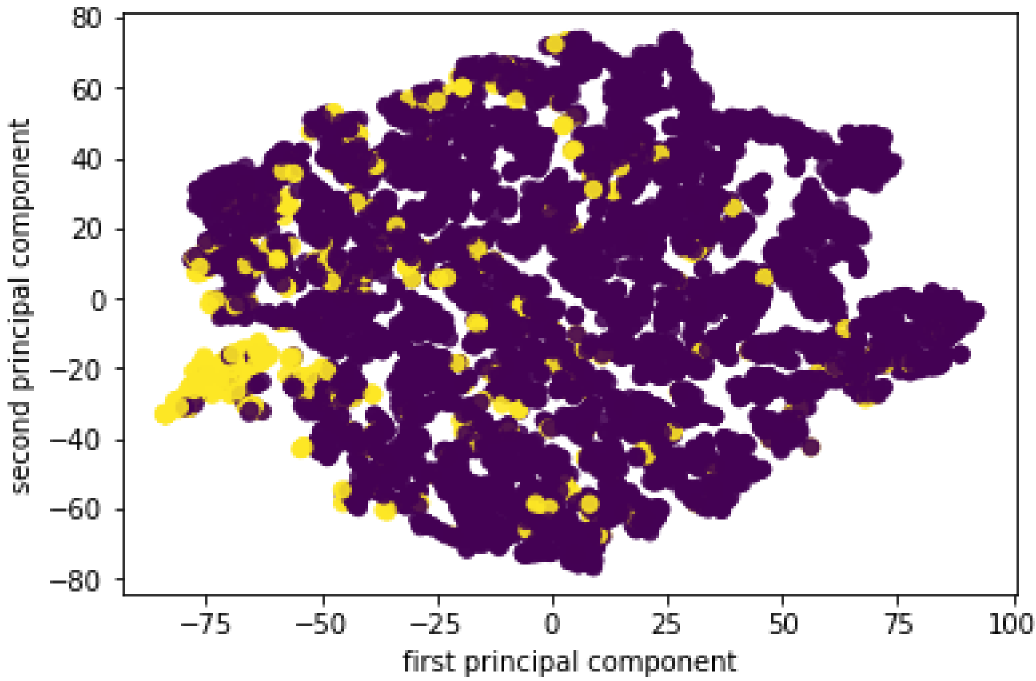
t-SNE with 2 components and tune the perplexity parameter equal to 300

```
In [89]: X_tsne = TSNE(n_components=2,perplexity=300).fit_transform(data)
         plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=target, alpha=.9)
         plt.xlabel("first principal component")
         plt.ylabel("second principal component")

Out[89]: Text(0,0.5,'second principal component')
```



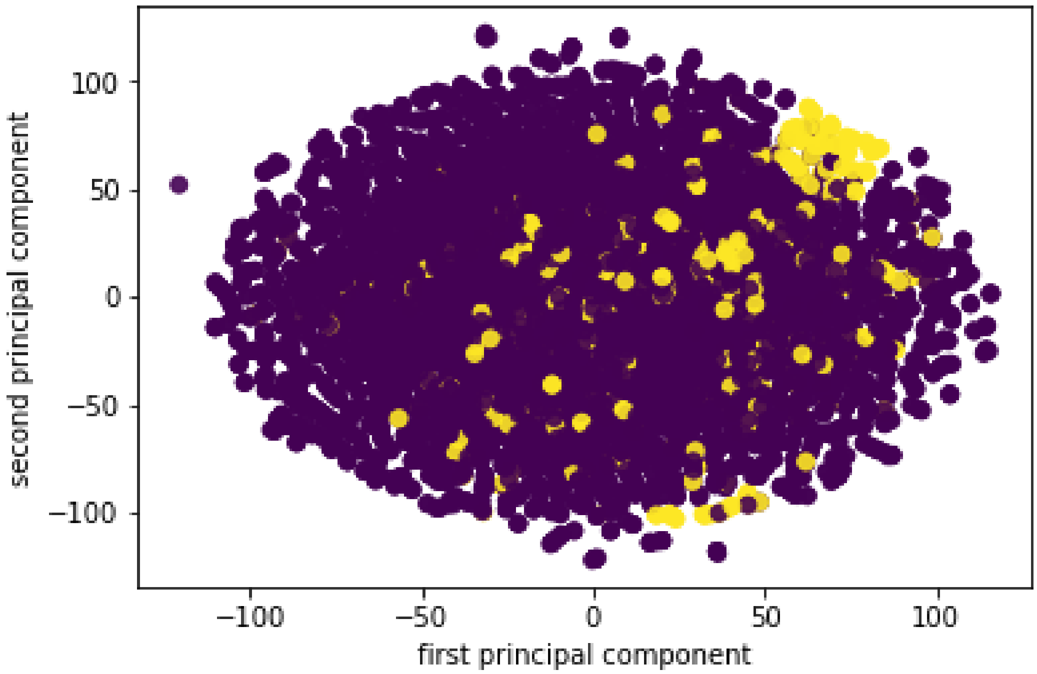
t-SNE scaling with 2 components

```
In [82]: from sklearn.pipeline import make_pipeline
         from sklearn.preprocessing import StandardScaler
         tsne_scaled = make_pipeline(StandardScaler(), TSNE(n_components=2))
         X_tsne_scaled = tsne_scaled.fit_transform(data)
         plt.scatter(X_tsne_scaled[:, 0], X_tsne_scaled[:, 1], c=target, alpha=.9)
         plt.xlabel("first principal component")
         plt.ylabel("second principal component")

Out[82]: Text(0,0.5,'second principal component')
```



t-SNE scaling with 2 components and tune the perplexity parameter equal to 2

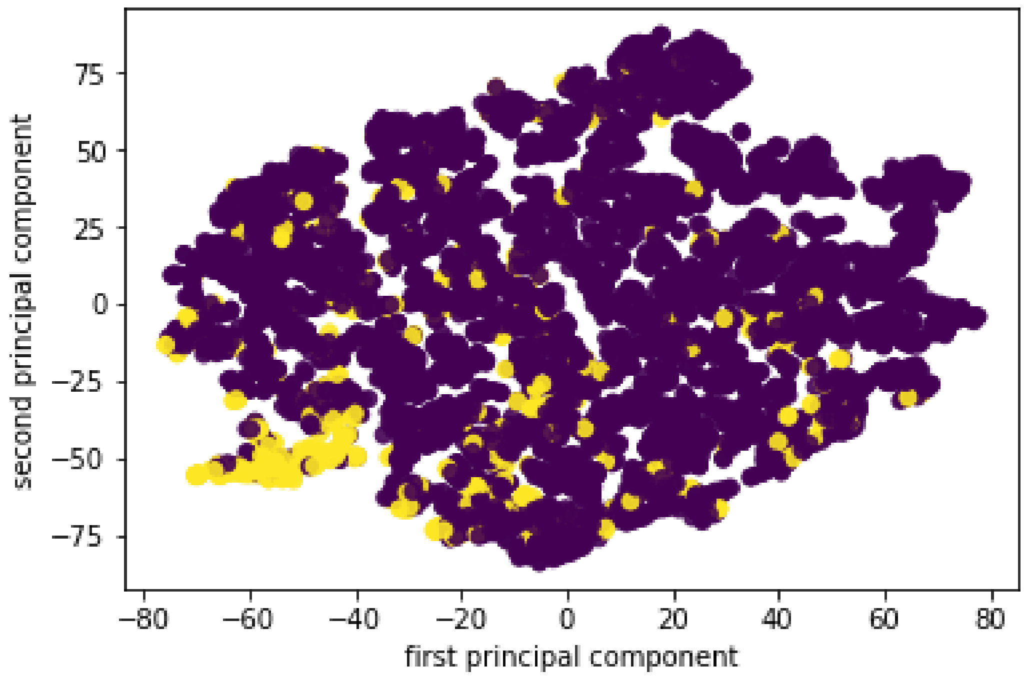
```
In [90]: tsne_scaled = make_pipeline(StandardScaler(), TSNE(n_components=2,perplexity=2))
         X_tsne_scaled = tsne_scaled.fit_transform(data)
         plt.scatter(X_tsne_scaled[:, 0], X_tsne_scaled[:, 1], c=target, alpha=.9)
         plt.xlabel("first principal component")
         plt.ylabel("second principal component")

Out[90]: Text(0,0.5,'second principal component')
```



t-SNE scaling with 2 components and tune the perplexity parameter equal to 30

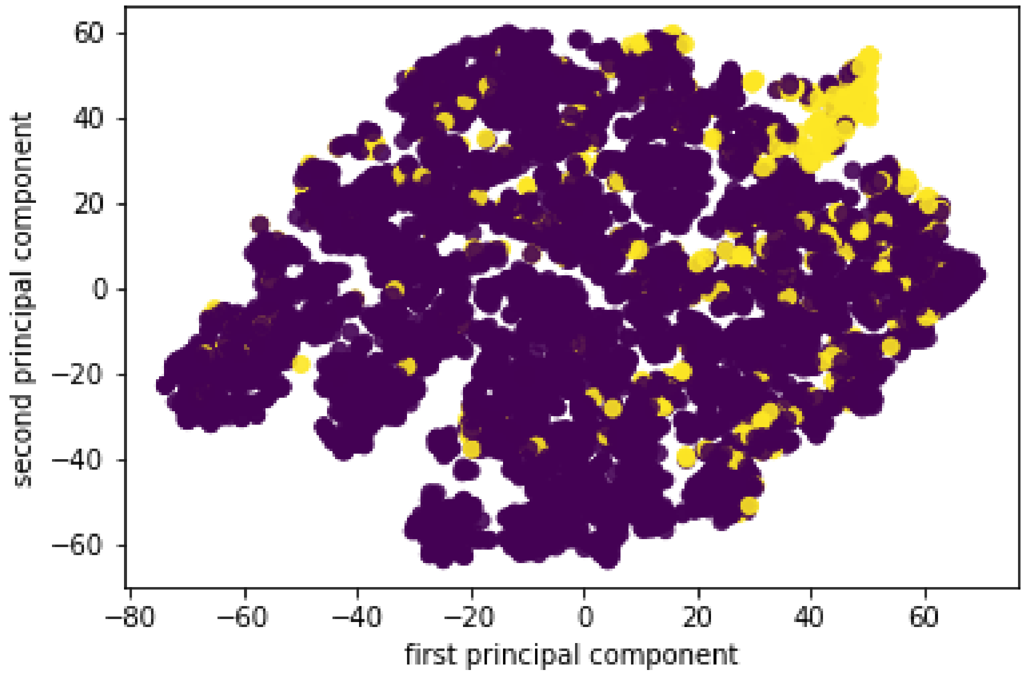
```
In [91]: tsne_scaled = make_pipeline(StandardScaler(), TSNE(n_components=2,perplexity=30))
         X_tsne_scaled = tsne_scaled.fit_transform(data)
         plt.scatter(X_tsne_scaled[:, 0], X_tsne_scaled[:, 1], c=target, alpha=.9)
         plt.xlabel("first principal component")
         plt.ylabel("second principal component")
```

```
Out[91]: Text(0,0.5,'second principal component')
```



t-SNE scaling with 2 components and tune the perplexity parameter equal to 50

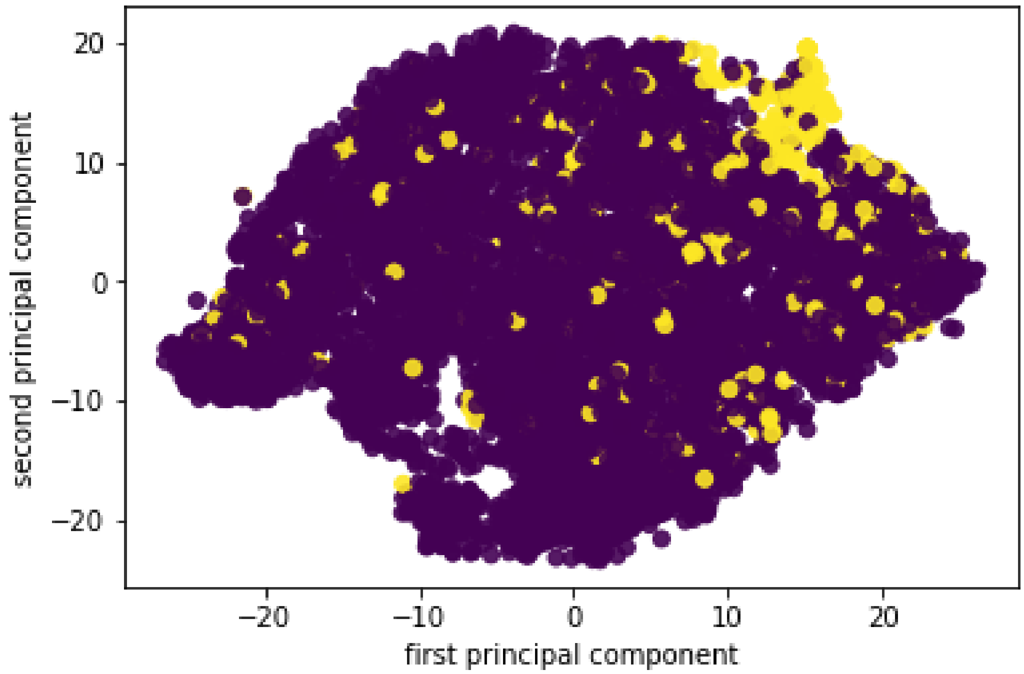
```
In [92]: tsne_scaled = make_pipeline(StandardScaler(), TSNE(n_components=2,perplexity=50))
         X_tsne_scaled = tsne_scaled.fit_transform(data)
         plt.scatter(X_tsne_scaled[:, 0], X_tsne_scaled[:, 1], c=target, alpha=.9)
         plt.xlabel("first principal component")
         plt.ylabel("second principal component")

Out[92]: Text(0,0.5,'second principal component')
```



t-SNE scaling with 2 components and tune the perplexity parameter equal to 300

```
In [93]: tsne_scaled = make_pipeline(StandardScaler(), TSNE(n_components=2,perplexity=300))
         X_tsne_scaled = tsne_scaled.fit_transform(data)
         plt.scatter(X_tsne_scaled[:, 0], X_tsne_scaled[:, 1], c=target, alpha=.9)
         plt.xlabel("first principal component")
         plt.ylabel("second principal component")
```

```
Out[93]: Text(0,0.5,'second principal component')
```



Answe: From T-sne with and without scaling, we can see that using T-sne is not good to obtain a better visualization

3 Problem 2

3.1 1.1

```
In [50]: # Defined by Professor Muller
         def silhouette_plot(X, cluster_labels,silhouette_scores,ax=None):
             #silhouette_scores = silhouette_samples(X, cluster_labels)
             if ax is None:
                 ax = plt.gca()
             y_lower = 10
             inliers = cluster_labels != -1
             X = X[inliers]
             cluster_labels = cluster_labels[inliers]
             silhouette_scores = silhouette_scores[inliers]
             labels = np.unique(cluster_labels)
             cm = plt.cm.Vega10 if len(labels) <= 10 else plt.cm.Vega20
             for i in labels:
                 # Aggregate the silhouette scores for samples belonging to
                 # cluster i, and sort them
                 ith_cluster_silhouette_values = \
                     silhouette_scores[cluster_labels == i]
```

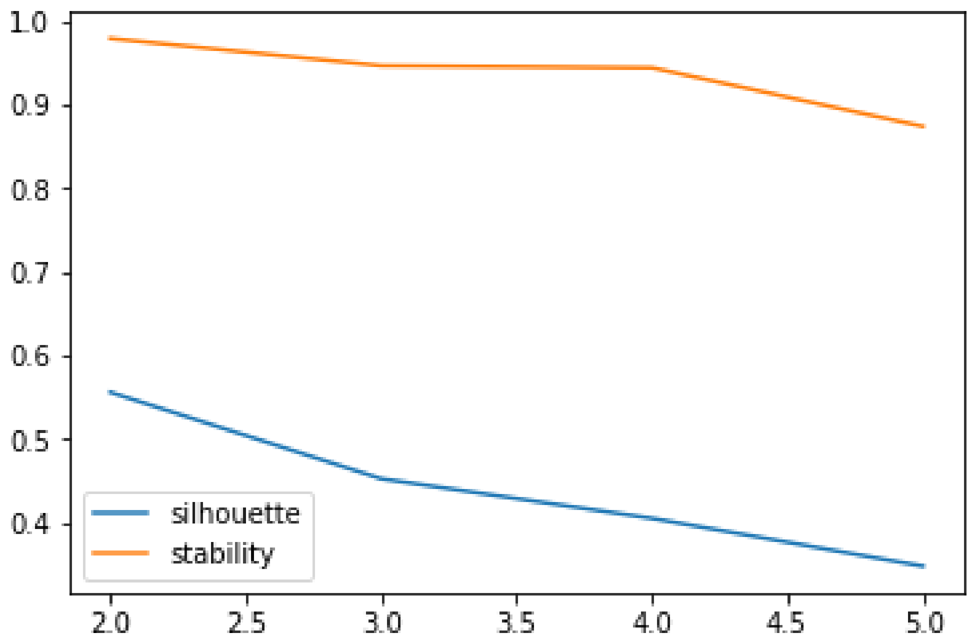
```
        ith_cluster_silhouette_values.sort()

        size_cluster_i = ith_cluster_silhouette_values.shape[0]
        y_upper = y_lower + size_cluster_i

        color = cm(i)
        ax.fill_betweenx(np.arange(y_lower, y_upper),
                          0, ith_cluster_silhouette_values,
                          facecolor=color, edgecolor=color, alpha=0.7)

        # Label the silhouette plots with their cluster numbers at the middle
        ax.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

        # Compute the new y_lower for next plot
        y_lower = y_upper + 10  # 10 for the 0 samples

from sklearn.base import clone
from sklearn.utils import check_random_state

def cluster_stability(X, est, n_iter=20, random_state=None):
    labels = []
    indices = []
    for i in range(n_iter):
        # draw bootstrap samples, store indices
        sample_indices = rng.randint(0, X.shape[0], X.shape[0])
        indices.append(sample_indices)
        est = clone(est)
        if hasattr(est, "random_state"):
            # randomize estimator if possible
            est.random_state = rng.randint(1e5)
        X_bootstrap = X[sample_indices]
        est.fit(X_bootstrap)
        # store clustering outcome using original indices
        relabel = -np.ones(X.shape[0], dtype=np.int)
        relabel[sample_indices] = est.labels_
        labels.append(relabel)
    scores = []
    for l, i in zip(labels, indices):
        for k, j in zip(labels, indices):
            # we also compute the diagonal which is a bit silly
            in_both = np.intersect1d(i, j)
            scores.append(adjusted_rand_score(l[in_both], k[in_both]))
    return np.mean(scores)
```

3.2 K-means

First, we can see how the silhouette score and stability changes according to different parameters.

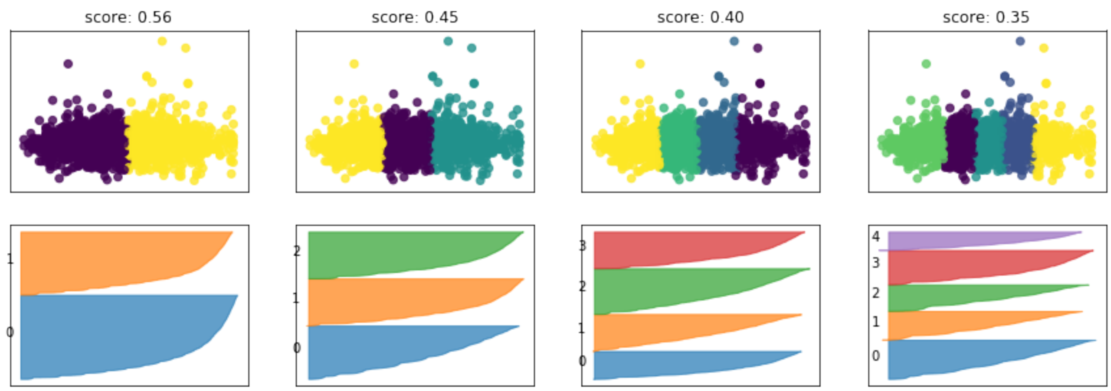
In [58]:
```python
X = HW4['X']
y = HW4['y'][:,0]

rng = np.random.RandomState(6)
silhouette,stability = [],[]
for i in range(2,6):
    km = KMeans(n_clusters=i) # not necessary to scale
    stability.append(cluster_stability(X, km))
    km.fit(X)
    silhouette.append(silhouette_score(X, km.labels_))

plt.plot(range(2,6),silhouette, label="silhouette")
plt.plot(range(2,6),stability, label="stability")
plt.legend()
```

Out[58]:
```
<matplotlib.legend.Legend at 0x10d7c8c88>
```



From this plot, 2 clusters may be the best choice. Then we inspect to the dataset.

In [63]:
```python
X_pca = PCA(n_components=2).fit_transform(X)
fig, axes = plt.subplots(2, 4, subplot_kw={'xticks': (), 'yticks':()}, figsize=(15, 5
for ax, n_clusters in zip(axes.T, [2,3,4,5]):
    km = KMeans(n_clusters=n_clusters)
    km.fit(X)
    ax[0].scatter(X_pca[:, 0], X_pca[:, 1], c=km.labels_, alpha=.8)
    silhouette_scores = silhouette_samples(X, km.labels_)
    silhouette_plot(X, km.labels_,silhouette_scores, ax=ax[1])
    ax[0].set_title("score: {:.2f}".format(silhouette_score(X, km.labels_)))
```

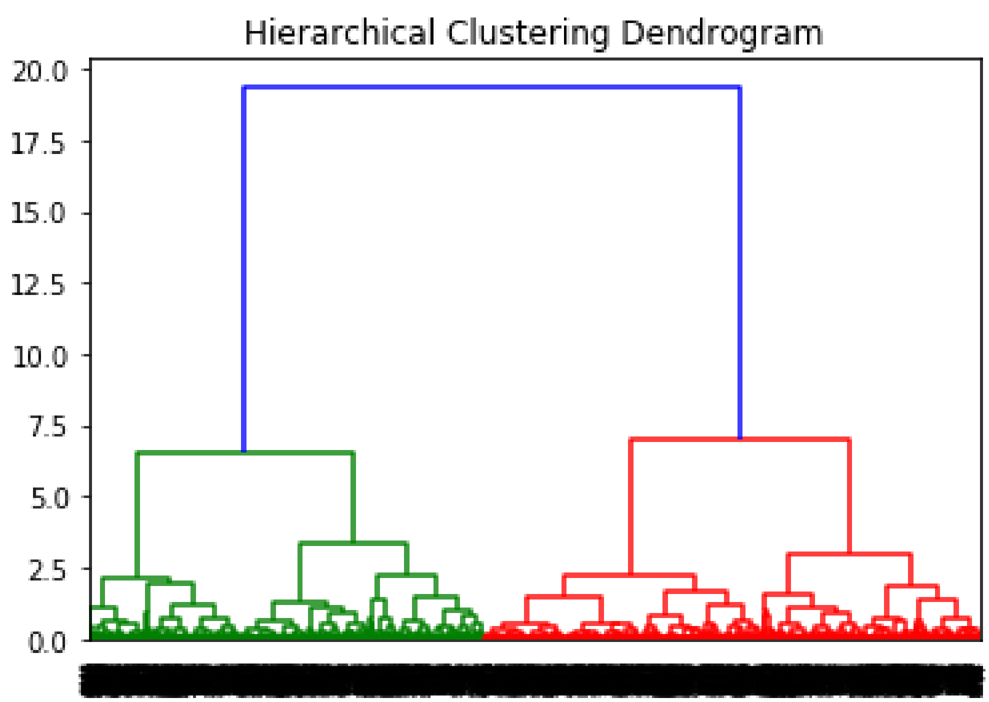


This also gives us same result, that 2 clusters is best.

3.3 Agglomerative Clustering

First, we could see the dendrogram for agglomerative clustering.

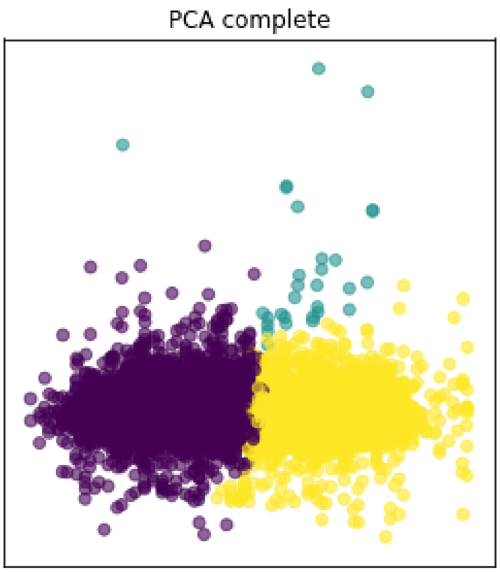
```
In [67]: from scipy.cluster.hierarchy import dendrogram, linkage

         linkage_matrix = linkage(X,'ward')
         dendrogram(linkage_matrix,truncate_mode="level",p=12,show_leaf_counts=True)
         plt.title('Hierarchical Clustering Dendrogram')
         plt.show()
```

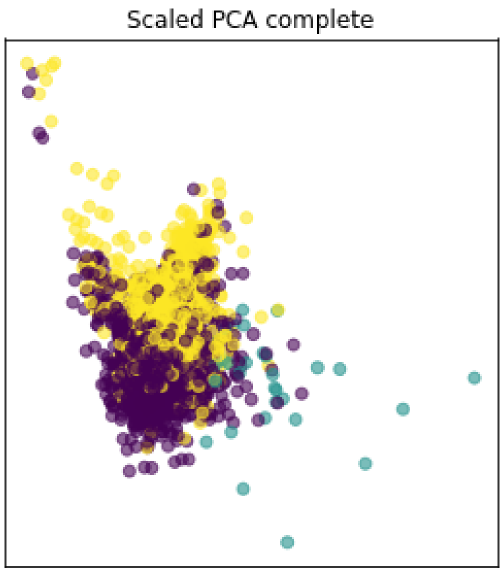


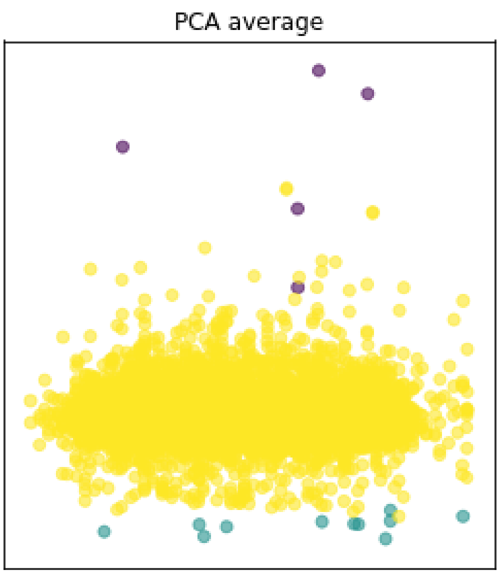
It seems like that 2 clusters is best. Then we can change the similarity criteria to choose best one.

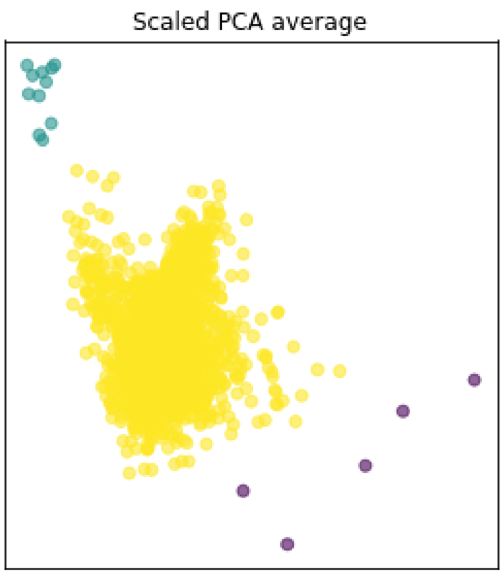
```
In [73]: fig, axes = plt.subplots(1, 2, subplot_kw={'xticks': (), 'yticks':()}, figsize=(10, 5
         pca_scaled = make_pipeline(StandardScaler(), PCA(n_components=2))
         X_pca_scaled = pca_scaled.fit_transform(X)
         X_pca=PCA(n_components=2).fit_transform(X)
         agg = AgglomerativeClustering(n_clusters=3, linkage='complete')
         agg.fit(X)
         axes[0].scatter(X_pca[:, 0], X_pca[:, 1], c=agg.labels_, alpha=.6)
         axes[0].set_title("PCA complete")
         axes[1].scatter(X_pca_scaled[:, 0], X_pca_scaled[:, 1], c=agg.labels_, alpha=.6)
         axes[1].set_title("Scaled PCA complete")

         fig, axes = plt.subplots(1, 2, subplot_kw={'xticks': (), 'yticks':()}, figsize=(10, 5
         pca_scaled = make_pipeline(StandardScaler(), PCA(n_components=2))
         X_pca_scaled = pca_scaled.fit_transform(X)
         X_pca=PCA(n_components=2).fit_transform(X)
         agg = AgglomerativeClustering(n_clusters=3, linkage='average')
         agg.fit(X)
         axes[0].scatter(X_pca[:, 0], X_pca[:, 1], c=agg.labels_, alpha=.6)
         axes[0].set_title("PCA average")
         axes[1].scatter(X_pca_scaled[:, 0], X_pca_scaled[:, 1], c=agg.labels_, alpha=.6)
         axes[1].set_title("Scaled PCA average")

         fig, axes = plt.subplots(1, 2, subplot_kw={'xticks': (), 'yticks':()}, figsize=(10, 5
         pca_scaled = make_pipeline(StandardScaler(), PCA(n_components=2))
         X_pca_scaled = pca_scaled.fit_transform(X)
         X_pca=PCA(n_components=2).fit_transform(X)
         agg = AgglomerativeClustering(n_clusters=3, linkage='ward')
         agg.fit(X)
         axes[0].scatter(X_pca[:, 0], X_pca[:, 1], c=agg.labels_, alpha=.6)
         axes[0].set_title("PCA ward")
         axes[1].scatter(X_pca_scaled[:, 0], X_pca_scaled[:, 1], c=agg.labels_, alpha=.6)
         axes[1].set_title("Scaled PCA ward")
```

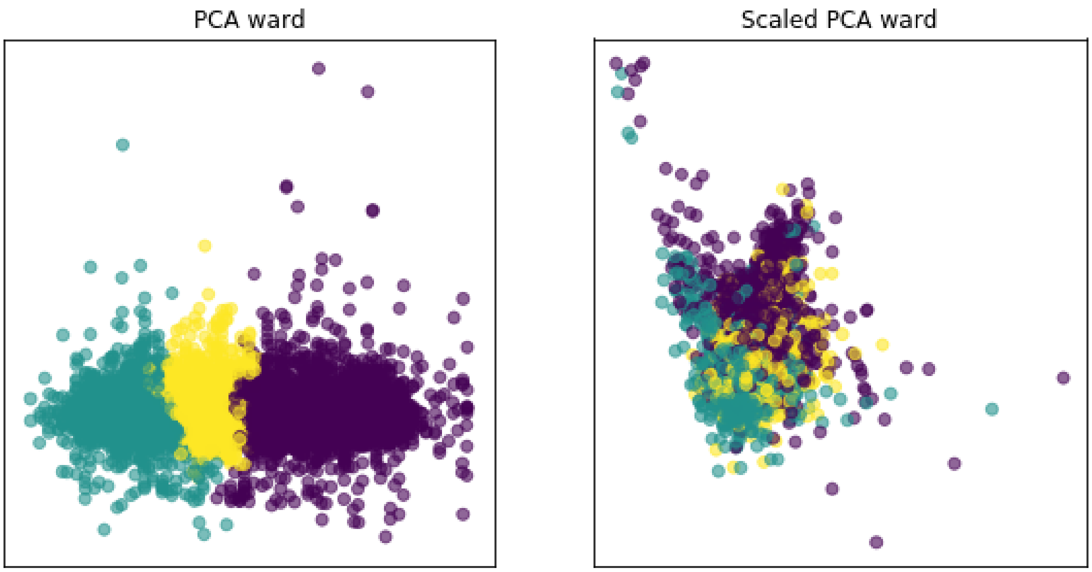
```
Out[73]: Text(0.5,1,'Scaled PCA ward')
```


PCA complete


Scaled PCA complete


PCA average

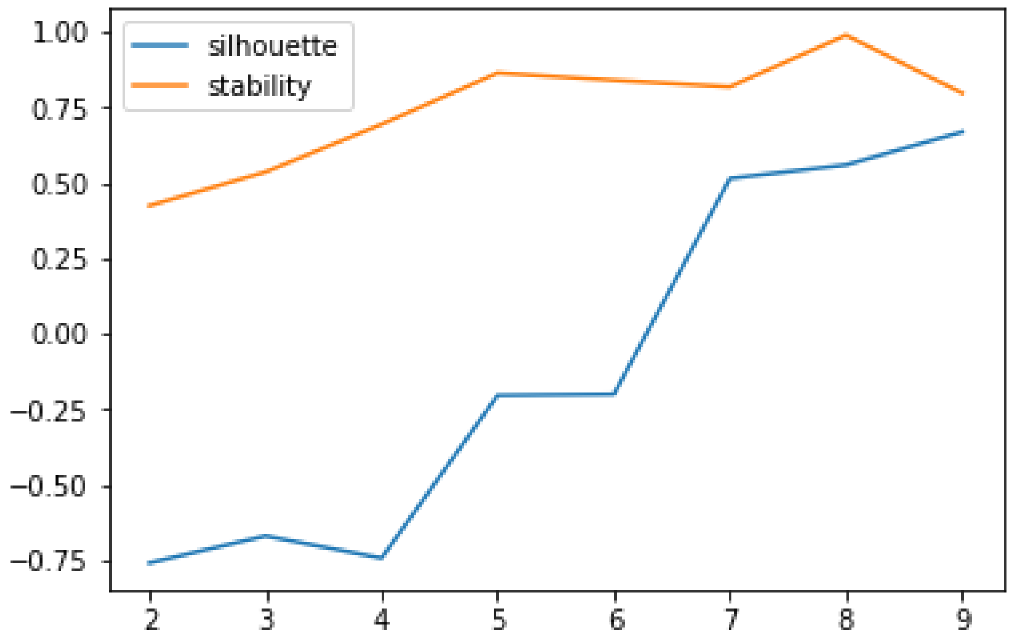

Scaled PCA average



From the plots, we can see when the similarity criteria is 'Average', it's best for outliers detection.

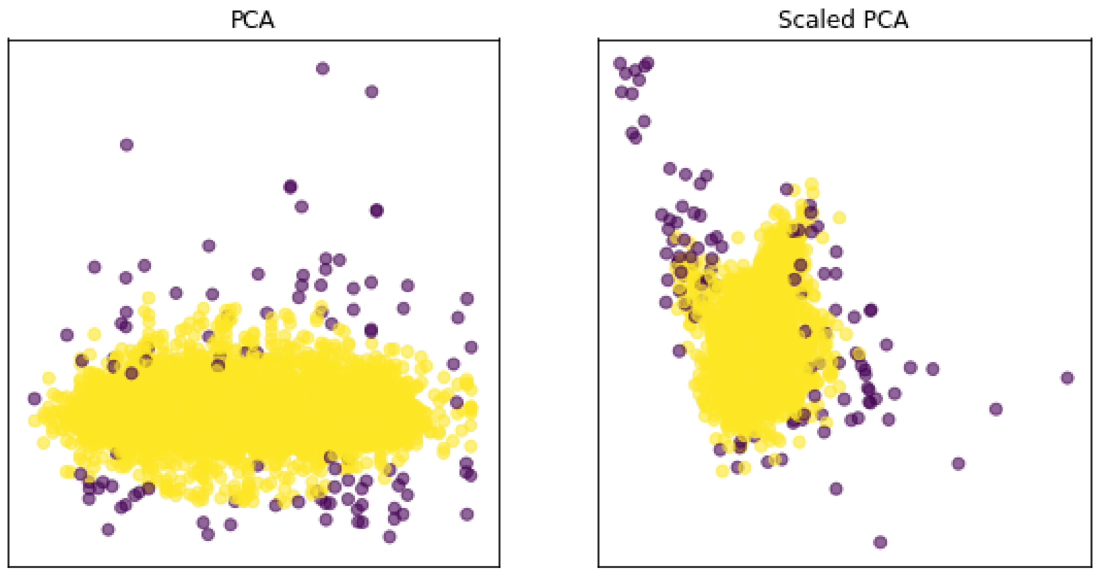
3.3.1 DBSCAN

```
In [74]: rng = np.random.RandomState(6)
         X = HW4['X']
         y = HW4['y'].flatten().astype(float)
         epses = np.logspace(-3, -.55, 10)
         sils,stability = [],[]
         for i in range(2,10):
             dbs = DBSCAN(eps=epses[i]) # not necessary to scale
             stability.append(cluster_stability(X, dbs))
             dbs.fit(X)
             sils.append(silhouette_score(X, dbs.labels_))

         plt.plot(range(2,10),sils, label="silhouette")
         plt.plot(range(2,10),stability, label="stability")
         plt.legend()

Out[74]: <matplotlib.legend.Legend at 0x1a1d5939e8>
```



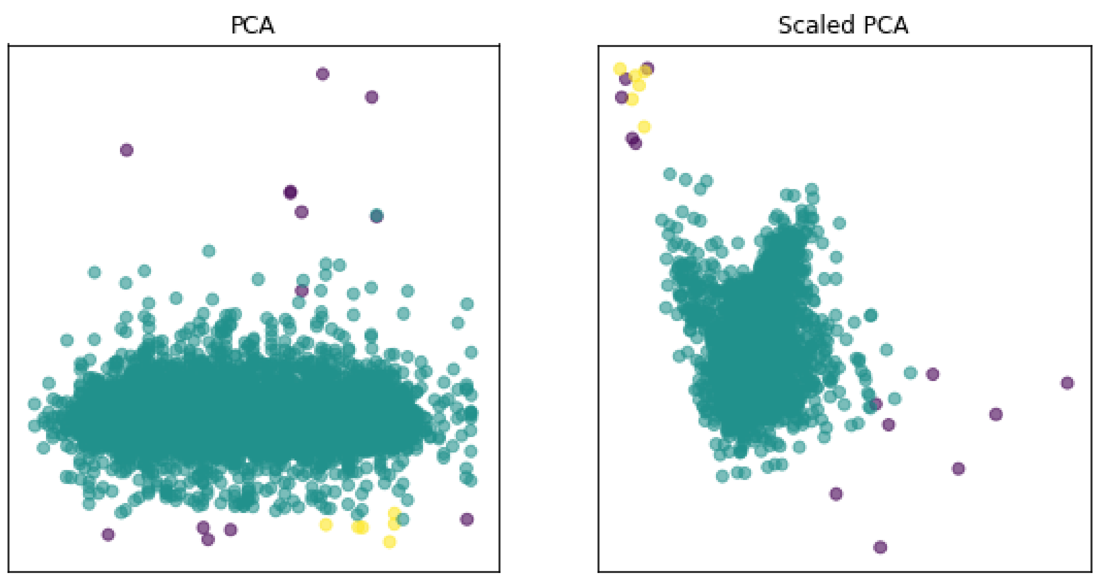
From the graph, we can see that when epsilon around 0.15058364, the stability is maximum and a high silhouette score. Then we drow PCA graph to show it.

```
In [75]: from sklearn.cluster import DBSCAN
         X_pca = PCA(n_components=2).fit_transform(X)
         eps = 0.05

         from sklearn.decomposition import PCA
         fig, axes = plt.subplots(1, 2, subplot_kw={'xticks': (), 'yticks':()}, figsize=(10, 5
         pca_scaled = make_pipeline(StandardScaler(), PCA(n_components=2))
         X_pca_scaled = pca_scaled.fit_transform(X)
         X_pca=PCA(n_components=2).fit_transform(X)
         dbs = DBSCAN(eps=eps).fit(X)# max
         axes[0].scatter(X_pca[:, 0], X_pca[:, 1], c=dbs.labels_, alpha=.6)
         axes[0].set_title("PCA")
         axes[1].scatter(X_pca_scaled[:, 0], X_pca_scaled[:, 1], c=dbs.labels_, alpha=.6)
         axes[1].set_title("Scaled PCA")

Out[75]: Text(0.5,1,'Scaled PCA')
```




```
In [76]: from sklearn.decomposition import PCA
         fig, axes = plt.subplots(1, 2, subplot_kw={'xticks': (), 'yticks':()}, figsize=(10, 5)
         pca_scaled = make_pipeline(StandardScaler(), PCA(n_components=2))
         X_pca_scaled = pca_scaled.fit_transform(X)
         X_pca=PCA(n_components=2).fit_transform(X)
         dbs = DBSCAN(eps=0.15).fit(X)# max
         axes[0].scatter(X_pca[:, 0], X_pca[:, 1], c=dbs.labels_, alpha=.6)
         axes[0].set_title("PCA")
         axes[1].scatter(X_pca_scaled[:, 0], X_pca_scaled[:, 1], c=dbs.labels_, alpha=.6)
         axes[1].set_title("Scaled PCA")
```

```
Out[76]: Text(0.5,1,'Scaled PCA')
```



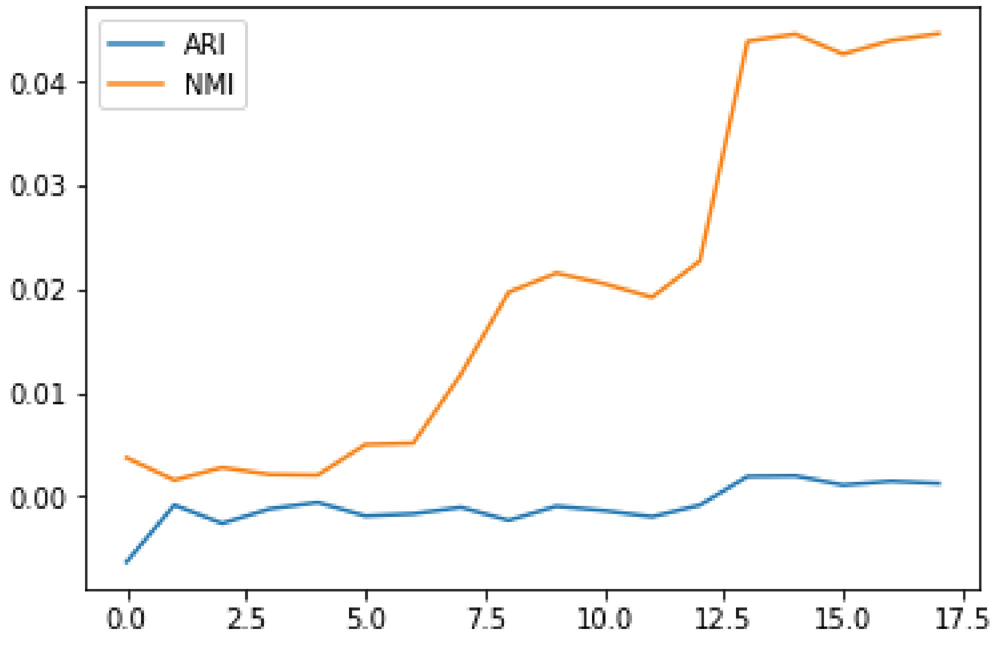
Then we can see that scaled PCA is much better than PCA, it much clear to see outliers.

3.4 2.2

3.4.1 Kmeans

In [77]:
```
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
from sklearn.neighbors import kneighbors_graph
from sklearn.metrics import adjusted_rand_score, silhouette_score, normalized_mutual_i
y = HW4['y'].flatten().astype(float)
aris, nmis = [],[]
for i in range(2,20):
    km = KMeans(n_clusters=i) # not necessary to scale
    km.fit(X)
    aris.append(adjusted_rand_score(y, km.labels_))
    nmi = normalized_mutual_info_score(y,km.labels_)
    nmis.append(nmi)

plt.plot(aris, label="ARI")

plt.plot(nmis, label="NMI")
# cluster size

plt.legend()
```

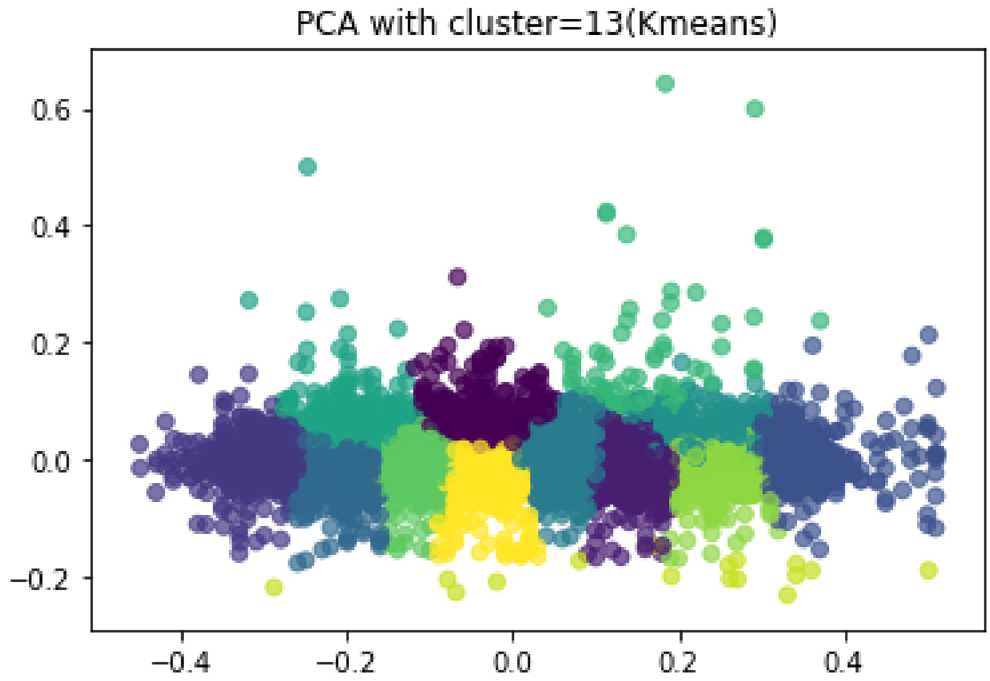
Out[77]: <matplotlib.legend.Legend at 0x1a1b3764e0>



Comapre with ARI and NMI score, we need choose n_cluster = 13

```
In [79]: X_pca = PCA(n_components=2).fit_transform(X)
         km = KMeans(n_clusters=13).fit(X)
         km.labels_
         plt.scatter(X_pca[:, 0], X_pca[:, 1], c=km.labels_, alpha=.7)
         plt.title("PCA with cluster=13(Kmeans)")
```

```
Out[79]: Text(0.5,1,'PCA with cluster=13(Kmeans)')
```




```
In [80]: print("adjusted_rand_score is {}".format(adjusted_rand_score(y,km.labels_)))
         print("normalized_mutual_info_score is {}".format(normalized_mutual_info_score(y,km.la
```

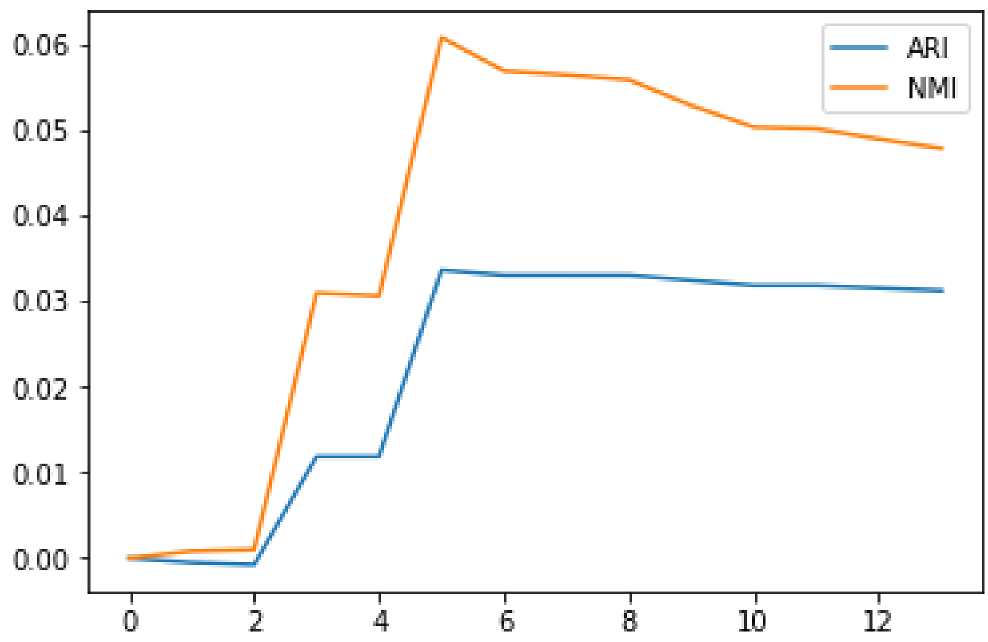
```
adjusted_rand_score is -0.0022930720248432747
normalized_mutual_info_score is 0.02010077358485314
```

3.4.2 Agglomerative Clustering

```
In [82]: from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
         from sklearn.neighbors import kneighbors_graph

         y = HW4['y'].flatten().astype(float)
```

```
aris, nmis= [],[]

for i in range(1,15):
    agg = AgglomerativeClustering(n_clusters=i, connectivity=lambda x: kneighbors_gra
    ari = adjusted_rand_score(y,agg.labels_)
    nmi = normalized_mutual_info_score(y,agg.labels_)
    aris.append(ari)
    nmis.append(nmi)


plt.plot(aris, label="ARI")
plt.plot(nmis, label="NMI")
plt.legend()
```

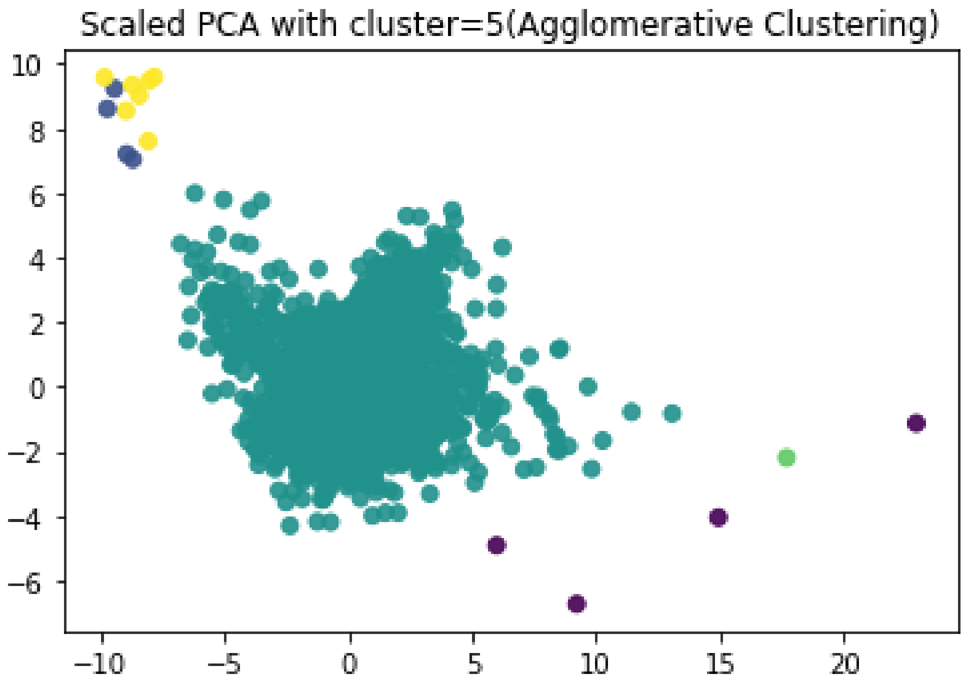
Out[82]: <matplotlib.legend.Legend at 0x1a1a6a3518>



From NMI and ARI score, we need to choose n_cluster=5.

```
In [83]: pca_scaled = make_pipeline(StandardScaler(), PCA(n_components=2))
         X_pca_scaled = pca_scaled.fit_transform(X)
         agg = AgglomerativeClustering(n_clusters=5, linkage='average')
         agg.fit(X)

         plt.scatter(X_pca_scaled[:, 0], X_pca_scaled[:, 1], c=agg.labels_, alpha=.9)
         plt.title("Scaled PCA with cluster=5(Agglomerative Clustering)")
```

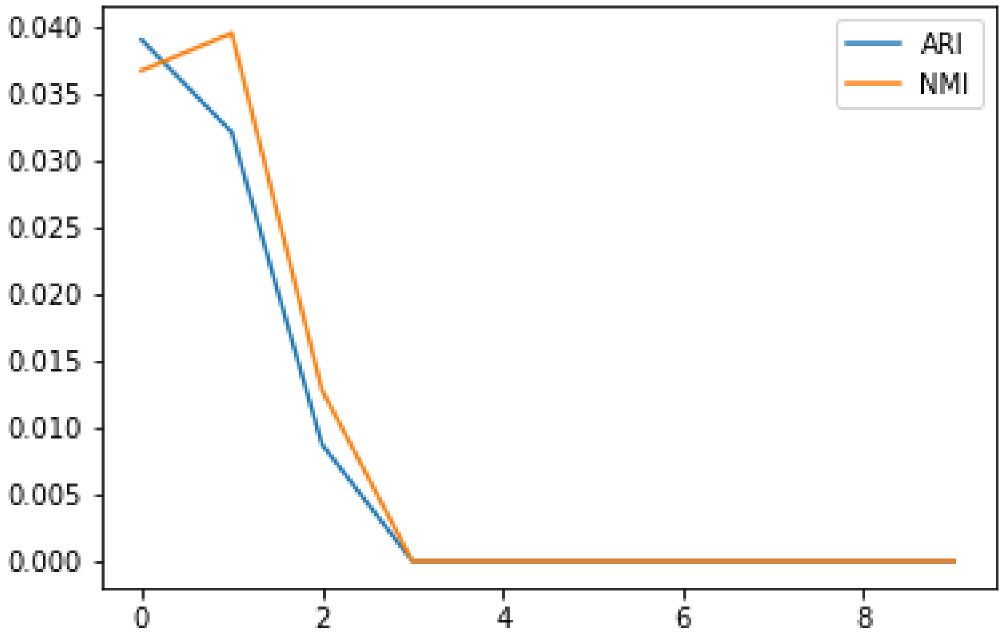
Out[83]: Text(0.5,1,'Scaled PCA with cluster=5(Agglomerative Clustering)')




```
In [84]: print("adjusted_rand_score is {}".format(adjusted_rand_score(y,agg.labels_)))
         print("normalized_mutual_info_score is {}".format(normalized_mutual_info_score(y,agg.l

adjusted_rand_score is 0.0330168126292234
normalized_mutual_info_score is 0.05776096679531449
```

3.4.3 DBSCAN

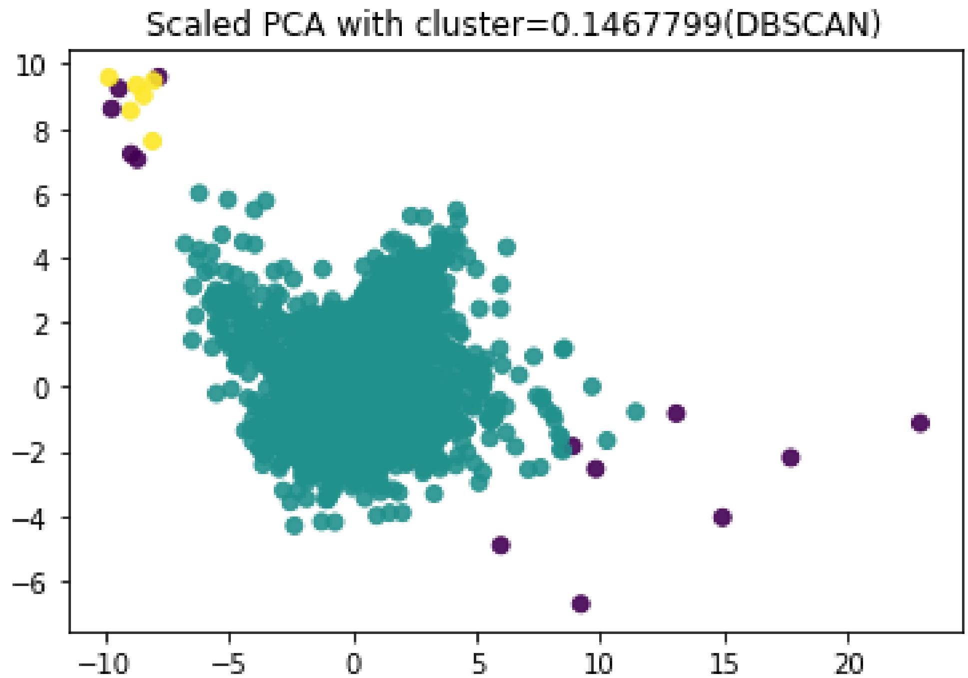
```
In [85]: aris, nmis = [],[]
         epses = np.logspace(-1, 0.5, 10)
         for i in range(10):
             dbs = DBSCAN(eps=epses[i]).fit(X) # not necessary to scale
             ari = adjusted_rand_score(y,dbs.labels_)
             nmi = normalized_mutual_info_score(y,dbs.labels_)
             aris.append(ari)
             nmis.append(nmi)


         plt.plot(aris, label="ARI")
         plt.plot(nmis, label="NMI")
         plt.legend()

Out[85]: <matplotlib.legend.Legend at 0x10dd73d68>
```




```
In [86]: np.logspace(-1, 0.5, 10)[1]

Out[86]: 0.14677992676220694
```

From NMI and ARI score, we need to choose n_cluster=0.146

```
In [90]: pca_scaled = make_pipeline(StandardScaler(), PCA(n_components=2))
         X_pca_scaled = pca_scaled.fit_transform(X)
         db = DBSCAN(eps=0.1467799).fit(X)
         plt.scatter(X_pca_scaled[:, 0], X_pca_scaled[:, 1], c=db.labels_, alpha=.9)
         plt.title("Scaled PCA with cluster=0.1467799(DBSCAN)")

Out[90]: Text(0.5,1,'Scaled PCA with cluster=0.1467799(DBSCAN)')
```




```
In [91]: print("adjusted_rand_score is {}".format(adjusted_rand_score(y,db.labels_)))
         print("normalized_mutual_info_score is {}".format(normalized_mutual_info_score(y,db.la
```

```
adjusted_rand_score is 0.03209877860896735
normalized_mutual_info_score is 0.03952681715971921
```

Answer: It is good.Even the NMI and ARI score is not very high, when we choose the highest NMI and ARI in every method, we can find outlier in PCA graph.

4 Problem 3

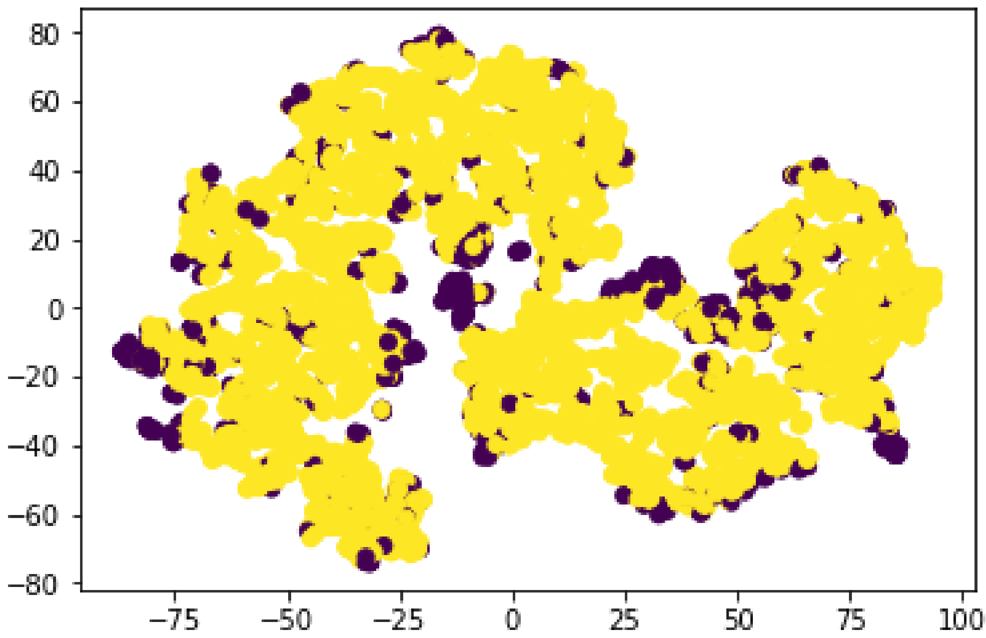
4.1 3.1

Assume the proportion of outliers is 10%.

```
In [7]: X=HW4['X']
        y=HW4['y']
        X_tsne = TSNE().fit_transform(X)
```

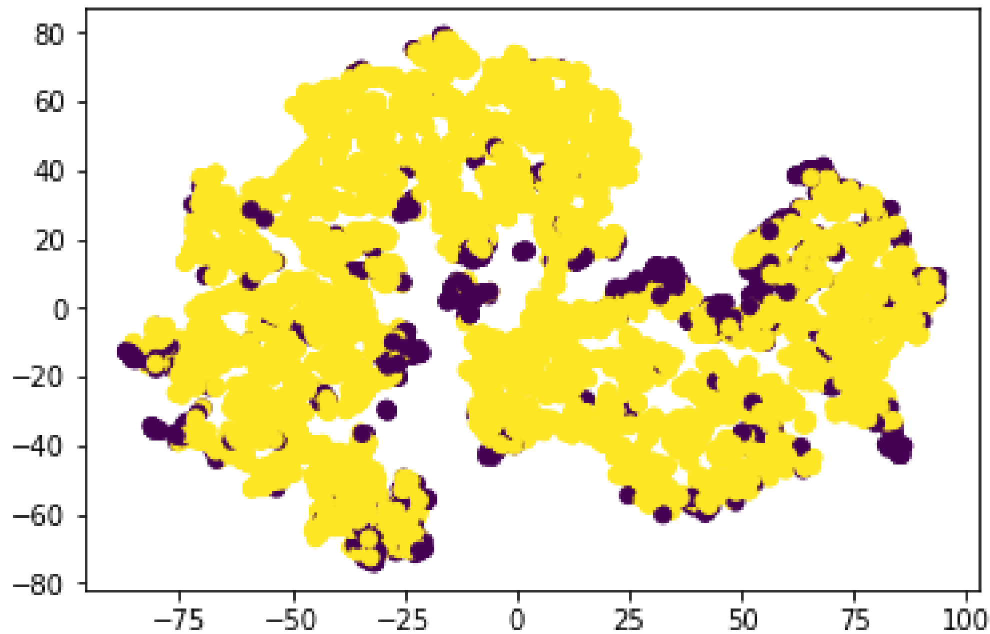
4.1.1 Elliptic Envelope

```
In [9]: ee = EllipticEnvelope(contamination=.1).fit(X)
        pred_ee = ee.predict(X)
        plt.scatter(X_tsne[:,0], X_tsne[:,1],c=pred_ee)
        plt.show()
```



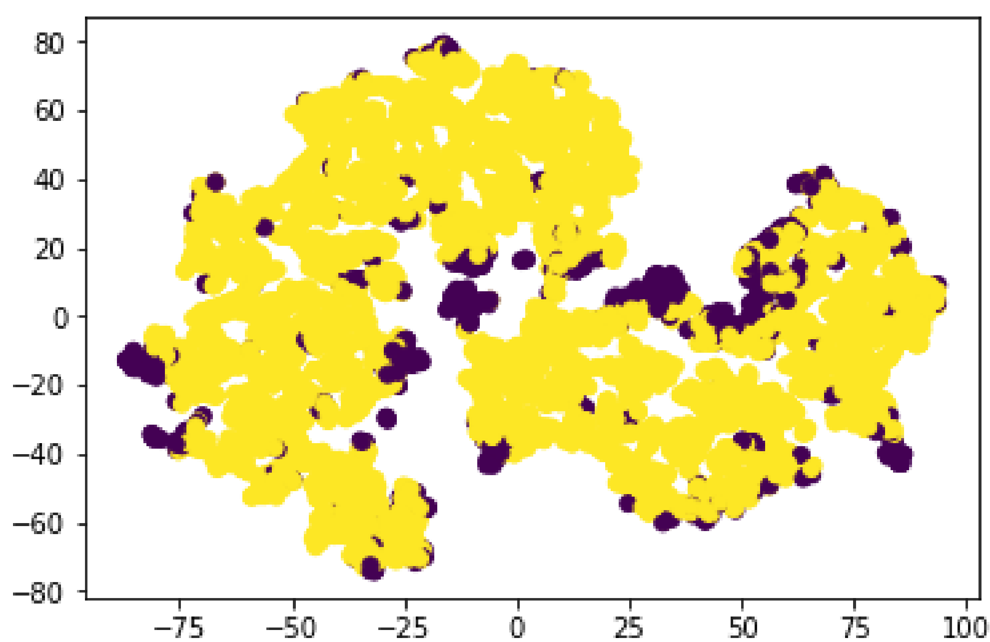
4.1.2 One Class SVM

```
In [10]: scaler = StandardScaler()
         X_scaled = scaler.fit_transform(X)
         oneclass = OneClassSVM(nu=.1).fit(X_scaled)
         pred_oneclass = oneclass.predict(X_scaled)
         plt.scatter(X_tsne[:,0], X_tsne[:,1],c=pred_oneclass)
         plt.show()
```



4.1.3 Isolation Forest

```
In [11]: isf = IsolationForest().fit(X)
         pred_isf = isf.predict(X)
         plt.scatter(X_tsne[:,0], X_tsne[:,1],c=pred_isf)
         plt.show()
```



From above plots, One Class SVM might have best perfermance since less predicted outliers appearing in the middle of class.

4.2 3.2

4.3 Using AUC and average precision to evaluate the different outlier detection approaches

4.3.1 Elliptic Envelope

```
In [46]: ap_ee = average_precision_score(y, -ee.decision_function(X))
         ee_auc = roc_auc_score(y, -ee.decision_function(X))
         ee_ari = adjusted_rand_score(y[:,0], pred_ee)
         ee_nmi = normalized_mutual_info_score(y[:,0], pred_ee)

         print("Average precision of Elliptic Envelope: {:.3f}".format(ap_ee))
         print("AUC for Elliptic Envelope: {:.3f}".format(ee_auc))
         print("ARI for Elliptic Envelope: {:.3f}".format(ee_ari))
         print("NMI for Elliptic Envelope: {:.3f}".format(ee_nmi))
```

```
Average precision of Elliptic Envelope: 0.504
AUC for Elliptic Envelope: 0.919
ARI for Elliptic Envelope: 0.351
NMI for Elliptic Envelope: 0.167
```

4.3.2 One Class SVM

```
In [47]: ap_oneclass = average_precision_score(y, -oneclass.decision_function(X))
         oneclass_auc = roc_auc_score(y, -oneclass.decision_function(X))
         oneclass_ari = adjusted_rand_score(y[:,0], pred_oneclass)
         oneclass_nmi = normalized_mutual_info_score(y[:,0], pred_oneclass)

         print("Average precision of One Class SVM: {:.3f}".format(ap_oneclass))
         print("AUC for One Class SVM: {:.3f}".format(oneclass_auc))
         print("ARI for One Class SVM: {:.3f}".format(oneclass_ari))
         print("NMI for One Class SVM: {:.3f}".format(oneclass_nmi))
```

```
Average precision of One Class SVM: 0.080
AUC for One Class SVM: 0.517
ARI for One Class SVM: 0.145
NMI for One Class SVM: 0.038
```

4.3.3 Isolation Forest

```
In [48]: ap_isf = average_precision_score(y, -isf.decision_function(X))
         isf_auc = roc_auc_score(y, -isf.decision_function(X))
```

```
isf_ari = adjusted_rand_score(y[:,0], pred_isf)
isf_nmi = normalized_mutual_info_score(y[:,0], pred_isf)

print("Average precision of Isolation Forest: {:.3f}".format(ap_isf))
print("AUC for Isolation Forest: {:.3f}".format(isf_auc))
print("ARI for One Class Isolation Forest: {:.3f}".format(isf_ari))
print("NMI for One Class Isolation Forest: {:.3f}".format(isf_nmi))
```

```
Average precision of Isolation Forest: 0.279
AUC for Isolation Forest: 0.787
ARI for One Class Isolation Forest: 0.207
NMI for One Class Isolation Forest: 0.069
```

Compared with clustering approaches in task2 by ARI and NMI, all three models are better.

5 Problem 4

```
In [17]: X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, random_state=0)
```

5.0.1 Undersampling

```
In [24]: lr_undersample_pipe = make_imb_pipeline(RandomUnderSampler(), StandardScaler(), Logis
         lr_scores = cross_validate(lr_undersample_pipe,X_train, y_train, cv=10, scoring=('roc
         rf_undersample_pipe = make_imb_pipeline(RandomUnderSampler(),RandomForestClassifier()
         rf_scores = cross_validate(rf_undersample_pipe, X_train, y_train, cv=10, scoring=('ro

         print('Logistic Regression AUC:{:.3f} Average Precision:{:.3f}'.format(lr_scores['tes
         print('Random Forest Classifier AUC:{:.3f} Average Precision:{:.3f}'.format(rf_scores
```

```
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
```

```
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
```

```
Logistic Regression AUC:0.952 Average Precision:0.696
Random Forest Classifier AUC:0.991 Average Precision:0.817
```

5.0.2 Oversampling

In [33]:
```python
lr_oversample_pipe = make_imb_pipeline(RandomOverSampler(), StandardScaler(), Logistic
lr_scores = cross_validate(lr_oversample_pipe,X_train, y_train, cv=10, scoring=('roc_a
rf_oversample_pipe = make_imb_pipeline(RandomOverSampler(),RandomForestClassifier())
rf_scores = cross_validate(rf_oversample_pipe, X_train, y_train, cv=10, scoring=('roc_

print('Logistic Regression AUC:{:.3f} Average Precision:{:.3f}'.format(lr_scores['test
print('Random Forest Classifier AUC:{:.3f} Average Precision:{:.3f}'.format(rf_scores
```

```
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
```

```
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
```

```
Logistic Regression AUC:0.987 Average Precision:0.817
Random Forest Classifier AUC:0.994 Average Precision:0.886
```

```
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
```

1. Based on the above results, select oversampling method.
2. Compare above results with the outlier detection in terms of AUC and average precision, using LogisticRegression and RandomForestClassifier is better.

5.1 Tune Parameter

5.1.1 Logistic Regression_Oversampling_GridSearch

```
In [31]: para_grid_lr={'logisticregression__C':[0.01,0.1,1,10,100]}
         grid_lr = GridSearchCV(lr_oversample_pipe, para_grid_lr)
         grid_lr.fit(X_train,y_train)
```

```
lr_auc_grid = roc_auc_score(y_test, grid_lr.predict_proba(X_test)[:, 1])
ap_lr_grid = average_precision_score(y_test, grid_lr.predict_proba(X_test)[:, 1])

print('best parameters for Logistic Regression: {}'.format(grid_lr.best_params_))
print("AUC for Logistic Regression: {:.3f}".format(lr_auc_grid))
print("Average precision of Logistic Regression: {:.3f}".format(ap_lr_grid))
```

```
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
```

```
best parameters for Logistic Regression: {'logisticregression__C': 10}
AUC for Logistic Regression: 0.991
Average precision of Logistic Regression: 0.847
```

```
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data(
  y = column_or_1d(y, warn=True)
```

5.1.2 Random Forest_Oversampling_GridSearch

In [34]:
```
para_grid_rf={'randomforestclassifier__max_depth': [2, 4, 8, 16], 'randomforestclassi
grid_rf = GridSearchCV(rf_oversample_pipe,para_grid_rf)
grid_rf.fit(X_train,y_train)

rf_auc_grid = roc_auc_score(y_test, grid_rf.predict_proba(X_test)[:, 1])
ap_rf_grid = average_precision_score(y_test, grid_rf.predict_proba(X_test)[:, 1])

print('best parameters for Random Forest Classifer: {}'.format(grid_rf.best_params_))
print("AUC for Random Forest Classifier: {:.3f}".format(rf_auc_grid))
print("Average precision of Random Forest Classifier: {:.3f}".format(ap_rf_grid))
```

```
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
```

```
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
```

```
best parameters for Random Forest Classifer: {'randomforestclassifier__max_depth': 16, 'random
AUC for Random Forest Classifier: 0.995
Average precision of Random Forest Classifier: 0.922
```

changing the class-weight to "balanced"

5.1.3 Logistic Regression_Oversampling_GridSearch_Balanced

In [35]:
```
lr_oversample_pipe_bal = make_imb_pipeline(RandomOverSampler(), StandardScaler(), Logi
para_grid_lr={'logisticregression__C':[0.01,0.1,1,10,100]}
grid_lr = GridSearchCV(lr_oversample_pipe_bal, para_grid_lr)
grid_lr.fit(X_train,y_train)

lr_auc_grid = roc_auc_score(y_test, grid_lr.predict_proba(X_test)[:, 1])
ap_lr_grid = average_precision_score(y_test, grid_lr.predict_proba(X_test)[:, 1])

print('best parameters for Logistic Regression: {}'.format(grid_lr.best_params_))
print("AUC for Logistic Regression: {:.3f}".format(lr_auc_grid))
print("Average precision of Logistic Regression: {:.3f}".format(ap_lr_grid))
```

```
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)


best parameters for Logistic Regression: {'logisticregression__C': 100}
AUC for Logistic Regression: 0.991
Average precision of Logistic Regression: 0.841


/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
```

```
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
```

5.1.4 Random Forest_Oversampling_GridSearch_Balanced

```
In [36]: rf_oversample_pipe_bal = make_imb_pipeline(RandomOverSampler(),RandomForestClassifier
         para_grid_rf={'randomforestclassifier__max_depth': [2, 4, 8, 16], 'randomforestclassi
         grid_rf = GridSearchCV(rf_oversample_pipe_bal,para_grid_rf)
         grid_rf.fit(X_train,y_train)

         rf_auc_grid = roc_auc_score(y_test, grid_rf.predict_proba(X_test)[:, 1])
         ap_rf_grid = average_precision_score(y_test, grid_rf.predict_proba(X_test)[:, 1])

         print('best parameters for Random Forest Classifer: {}'.format(grid_rf.best_params_))
         print("AUC for Random Forest Classifier: {:.3f}".format(rf_auc_grid))
         print("Average precision of Random Forest Classifier: {:.3f}".format(ap_rf_grid))
```

```
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
```

```
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
/Users/albertzhang/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
```

```
best parameters for Random Forest Classifer: {'randomforestclassifier__max_depth': 16, 'random
AUC for Random Forest Classifier: 0.996
Average precision of Random Forest Classifier: 0.930
```

According to the results, changing the class weight nearly doesn't help.