Name(s): _____

Homework 5: CSCI 347: Data Mining

Show your work. Include any code snippets you used to generate an answer, using comments in the code to clearly indicate which problem corresponds to which code.

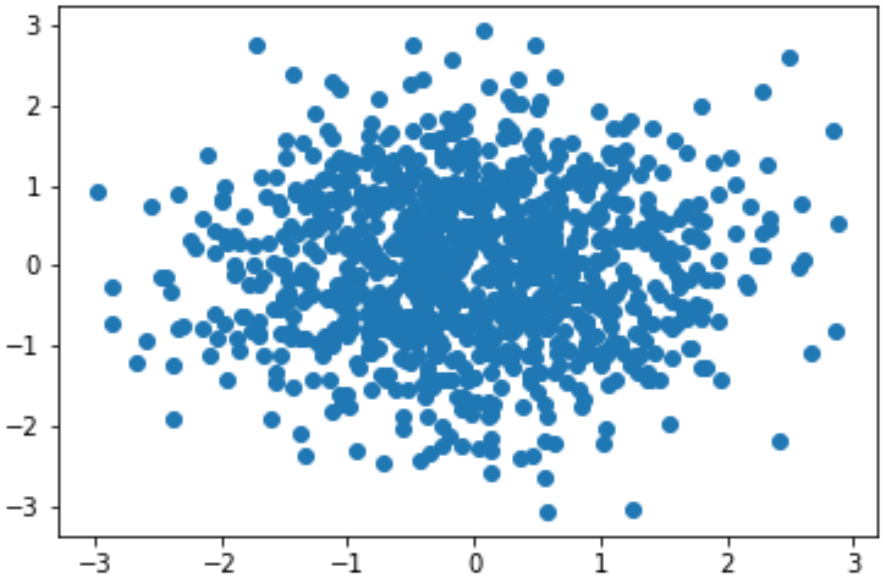
1. [2 points] In Python, generate a (2-dimensional multivariate Gaussian) data matrix D using the following code:

```
mu = np.array([0,0])
Sigma = np.array([[1,0], [0, 1]])
X1, X2 = np.random.multivariate_normal(mu, Sigma, 1000).T
D = np.array([X1, X2]).T
```

Create a scatter plot of the data, with the x-axis corresponding to the first attribute (column) in D, and the y-axis corresponding to the second attribute (column) in D.

Using the following code, we can create the plot below:

```
import matplotlib.pyplot as plt
plt.scatter(D[:,0],D[:,1])
plt.savefig('hw5-plot-q1.png')
```



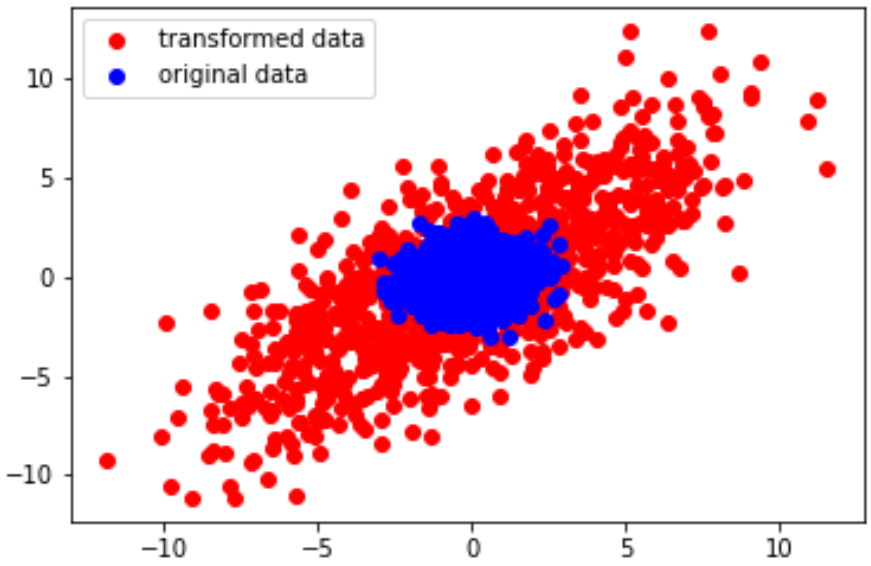
2. [7 points] Using the scaling matrix S and rotation matrix R below to transform the data D from Question 1, by multiplying each data instance (row) x_i by RS. Let RSD be the matrix of the transformed data. That is, each 2-dimensional row vector x_i in D should be transformed into a 2-dimensional vector RSx_i in RSD.

A. [4 points] Plot the transformed data RSD in the same figure as the original data D, using different colors to differentiate between the original and transformed data.

$$R = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}, \text{ where } \theta = \frac{\pi}{4}, S = \begin{pmatrix} 5 & 0 \\ 0 & 2 \end{pmatrix}$$

Using the following code, we obtain the plot below:

```
theta = np.pi/4
R = np.array([[np.cos(theta), -np.sin(theta)], [np.sin(theta), np.cos(theta)]])
S = np.array([[5,0],[0,2]])
RSD = R.dot(S.dot(D.T))
plt.scatter(RSD[0,:],RSD[1,:], c='r',label='transformed data')
plt.scatter(D[:,0],D[:,1],c='b',label='original data')
plt.legend(loc='upper left')
plt.savefig('transformed-data-hw5-q2.png')
```



B. [2 points] Write down the covariance matrix of the transformed data RSD.

Continuing from the code above, we can use numpy's cov function to find the covariance matrix:

np.cov(RSD)

produces the matrix:

$$\begin{pmatrix} 14.4614 & 10.7904 \\ 10.7904 & 14.8703 \end{pmatrix}$$

C. [1 point] What is the total variance of the transformed data RSD?

Summing along the diagonal of the covariance matrix above gives us the total variance:

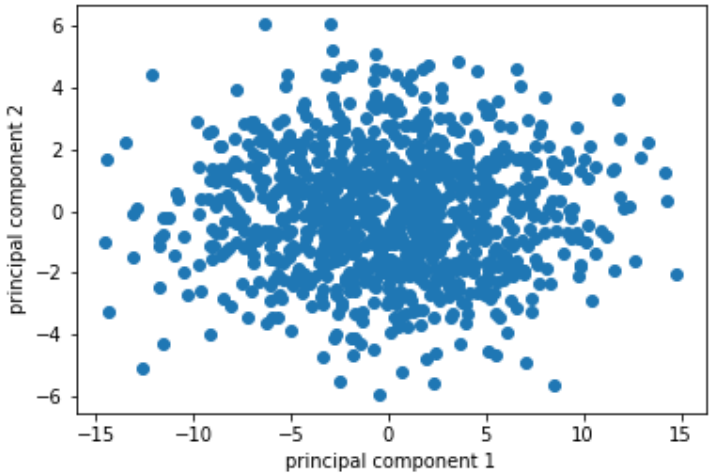
14.4614 + 14.8703 = 29.3317

3. [8 points] Use sklearn's PCA function to transform the data matrix RSD from Question 2 to a 2-dimensional space where the coordinate axes are the principal components.

   A. [4 points] Plot the PCA-transformed data, with the x-axis corresponding to the first principal component and the y-axis corresponding to the second principal component.

We can use the following code to produce the plot:

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pcaRSD = pca.fit_transform(RSD.T)
plt.scatter(pcaRSD[:,0], pcaRSD[:,1])
plt.xlabel('principal component 1')
plt.ylabel('principal component 2')
plt.savefig('pcaRSD-hw5-q3.png')
```



B. [2 points] What is the (sample) covariance matrix of the PCA-transformed data?

Continuing with the code above, using np.cov(pcaRSD.T), we get the following covariance matrix:

$$\begin{pmatrix} 25.4581 & 0 \\ 0 & 3.8735 \end{pmatrix}$$

C. [2 points] What is the fraction of the total variance captured in the direction of the first principal component? What is the fraction of the total variance captured in the direction of the second principal component?

Continuing with the code above, we can use:

pca.explained_variance_ratio_

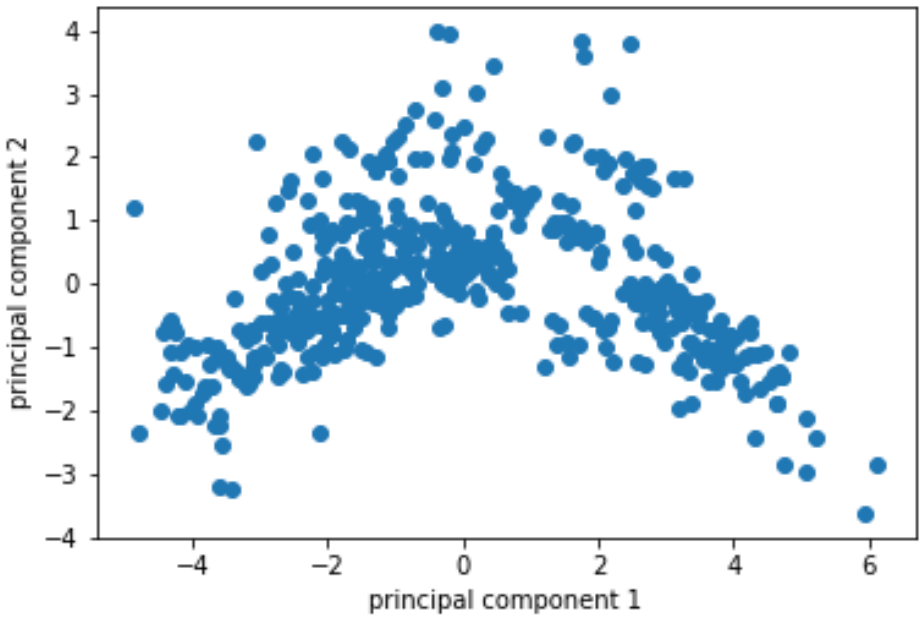
To see that 86.8% of the total variance is captured in the direction of the first principal component, and 13.2% of the total variance is captured in the direction of the second principal component.

4. [18 points] Load the boston data set into Python using sklearn's **datasets** package. Use sklearn's PCA function to reduce the dimensionality of the data to 2 dimensions.

    1. [5 points] First, standard-normalize the data. Then, create a scatter plot of the 2-dimensional, PCA-transformed normalized Boston data, with the x-axis corresponding to the first principal component and the y-axis corresponding to the second principal component.

We can use the following code to produce the requested plot that is pictured below:

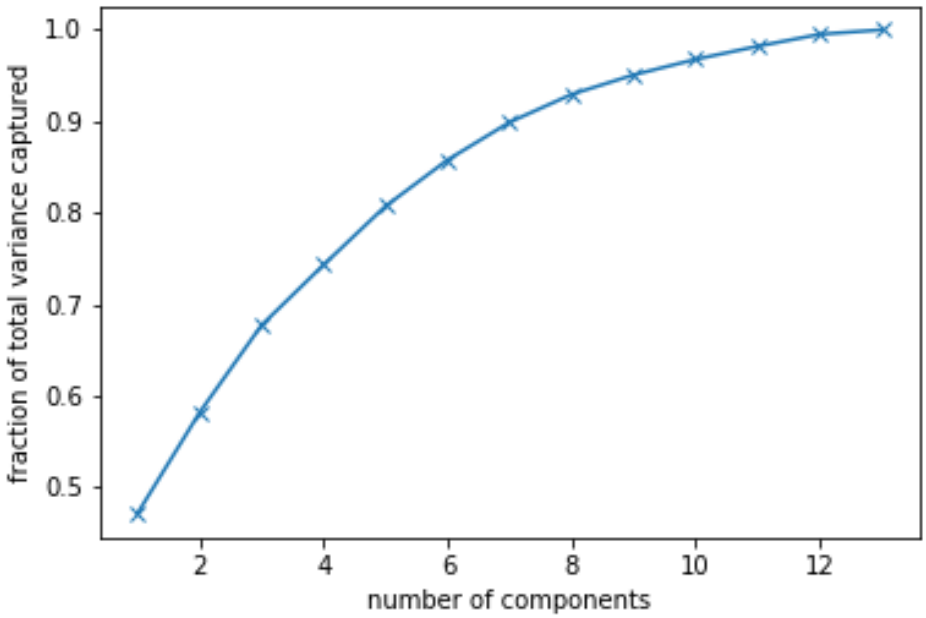
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_normalized = sc.fit_transform(X)
pca = PCA(n_components=2)
pcaXn = pca.fit_transform(X_normalized)
plt.scatter(pcaXn[:,0], pcaXn[:,1])
plt.xlabel('principal component 1')
plt.ylabel('principal component 2')
plt.savefig('boston-pca-plot-hw-q4.png')
```



2. [3 points] Create a plot of the fraction of the total variance explained by the first r components for $r = 1,\ldots,13$.

The following code will produce the requested plot, pictured below:

```
pca = PCA()
pcaXn = pca.fit_transform(X_normalized)
plt.plot(range(1,14), np.cumsum(pca.explained_variance_ratio_),marker='x')
plt.xlabel('number of components')
plt.ylabel('fraction of total variance captured')
plt.savefig('frac-total-variance-boston-normalized-data-hw5-q3.png')
```



3. [2 points]
    1. [1 point] If we want to capture at least 90% of the variance of the normalized Boston data, how many principal components (i.e., what dimensionality) should we use?

We would need to use at least 8 principal components, which capture 92.95% of the total variance. Using 7 would only capture 89.91%.

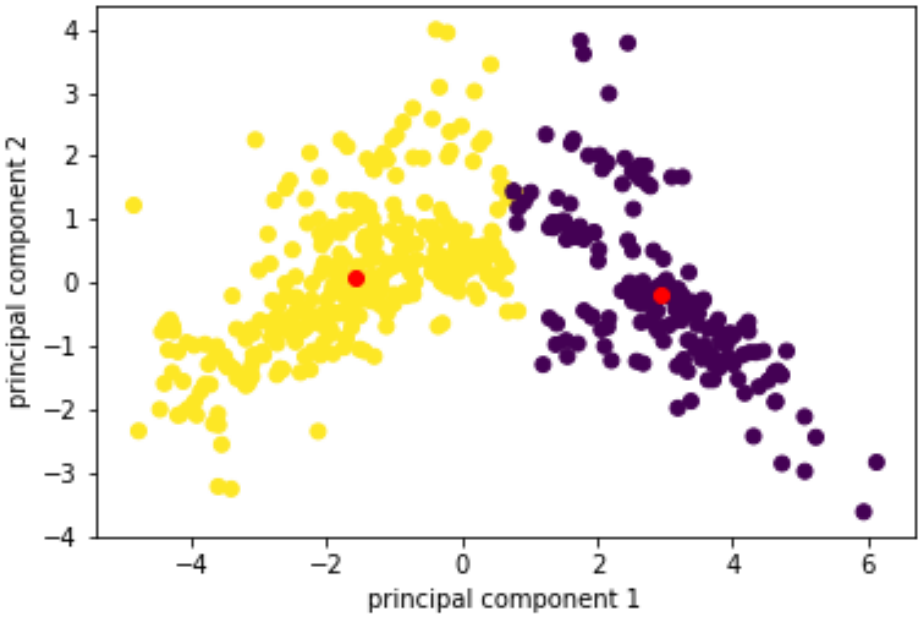
    2. [1 point] If we use two principal components of the normalized Boston data, how much (what fraction or percentage) of the total variance do we capture?

Two principal components capture 58.15% of the total variance.

4. [4 points] Use scikit-learn's implementation of k-means to find 2 clusters in the two-dimensional, PCA-transformed normalized Boston data set (the input to k-means should be the data that was plotted in part 4.1). Plot the 2-dimensional data with colors corresponding to predicted cluster membership for each point. On the same plot, also plot the the two means found by the k-means algorithm in a different color than the colors used for the data.

The following code will produce the requested plot below

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=2).fit(pcaXn)
plt.scatter(pcaXn[:,0], pcaXn[:,1],c=kmeans.labels_)
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1],c='r')
plt.xlabel('principal component 1')
plt.ylabel('principal component 2')
plt.savefig('kmeans-clustered-pca-boston-data-normalized.png')
```



5. [4 points] Use scikit-learn's implementation of DBSCAN to find clusters in the two-dimensional, PCA-transformed normalized Boston data set (the input to DBSCAN should be the data that was plotted in part 4.1). Plot the 2-dimensional data with colors corresponding to predicted cluster membership for each point. Noise points should be colored differently than any of the clusters. How many clusters were found by DBSCAN?

We can use the following code to find the clusters using DBSCAN:

```
from sklearn.cluster import DBSCAN
db = DBSCAN(eps=2, min_samples=4).fit(pcaXn)
```

We can examine how many clusters are found by printing the set of labels that is output by DBSCAN:
print(set(db.labels_))

The code above shows there are 7 clusters found using the parameter settings eps=2 and min_samples=4.

The code below will plot the clusters and noise points:

```
from sklearn.cluster import DBSCAN
db = DBSCAN(eps=2, min_samples=4).fit(pcaXn)
print(set(db.labels_))

plt.scatter(pcaXn[:,0], pcaXn[:,1],c=db.labels_)
plt.xlabel('principal component 1')
plt.ylabel('principal component 2')
plt.savefig('dbscan-clustered-pca-boston-data-normalized.png')
```