CSCI 347
Homework 05

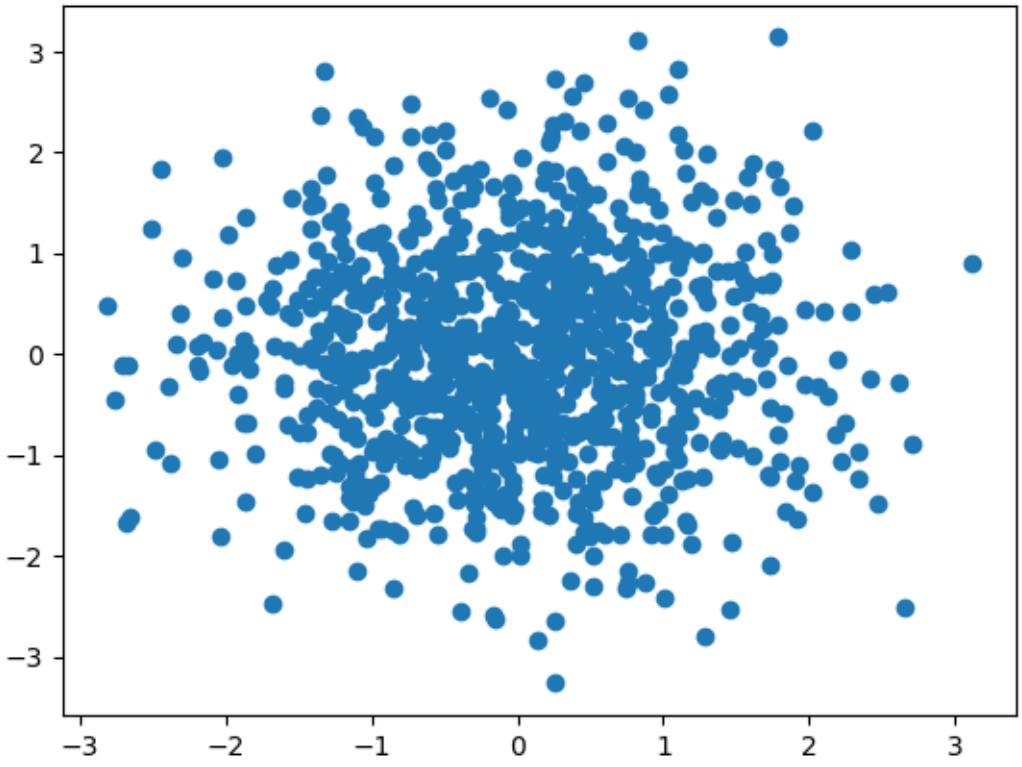
Show your work. Include any code snippets you used to generate an answer, using comments in the code to clearly indicate which problem corresponds to which code

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.

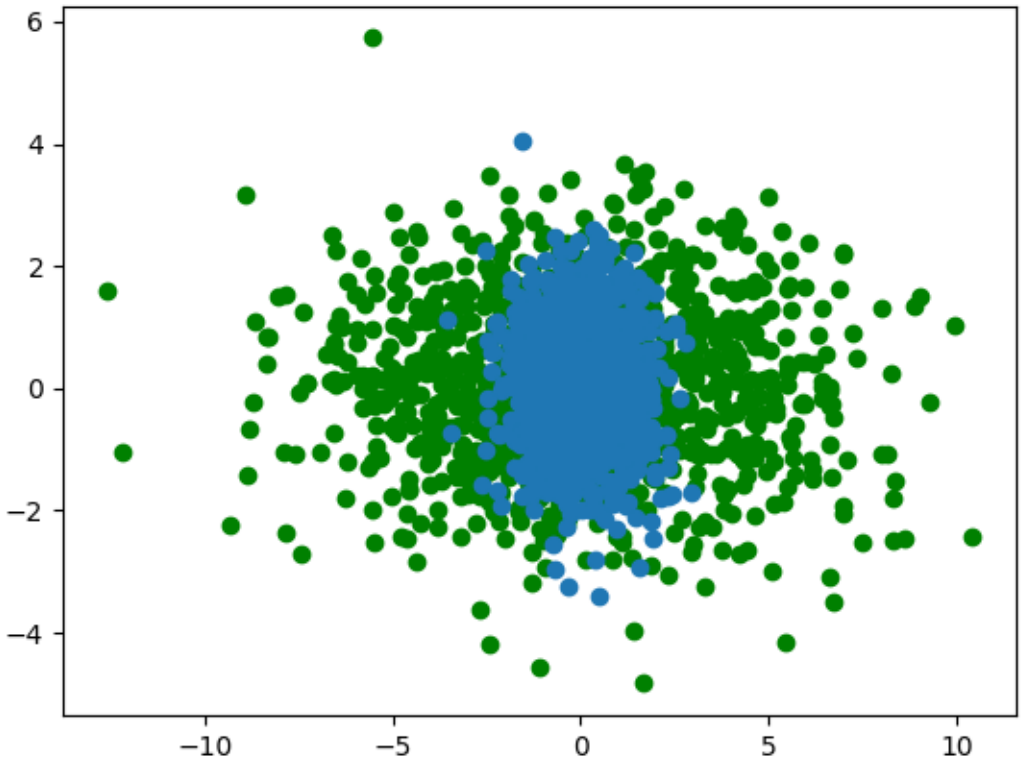
```
plt.scatter(D[:, 0], D[:, 1])
```



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 D_{RS} 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 D_{RS}.

   (a) (4 points) Plot the transformed data D_{RS} in the same figure as the original data D, using different colors to differentiate between the original and transformed data.

$$R = \begin{pmatrix} \cos(\pi/4) & -\sin(\pi/4) \\ \sin(\pi/4) & \cos(\pi/4) \end{pmatrix}, \qquad S = \begin{pmatrix} 5 & 0 \\ 0 & 2 \end{pmatrix}$$



(b) (2 points) Write down the covariance matrix of the transformed data D_{RS}.

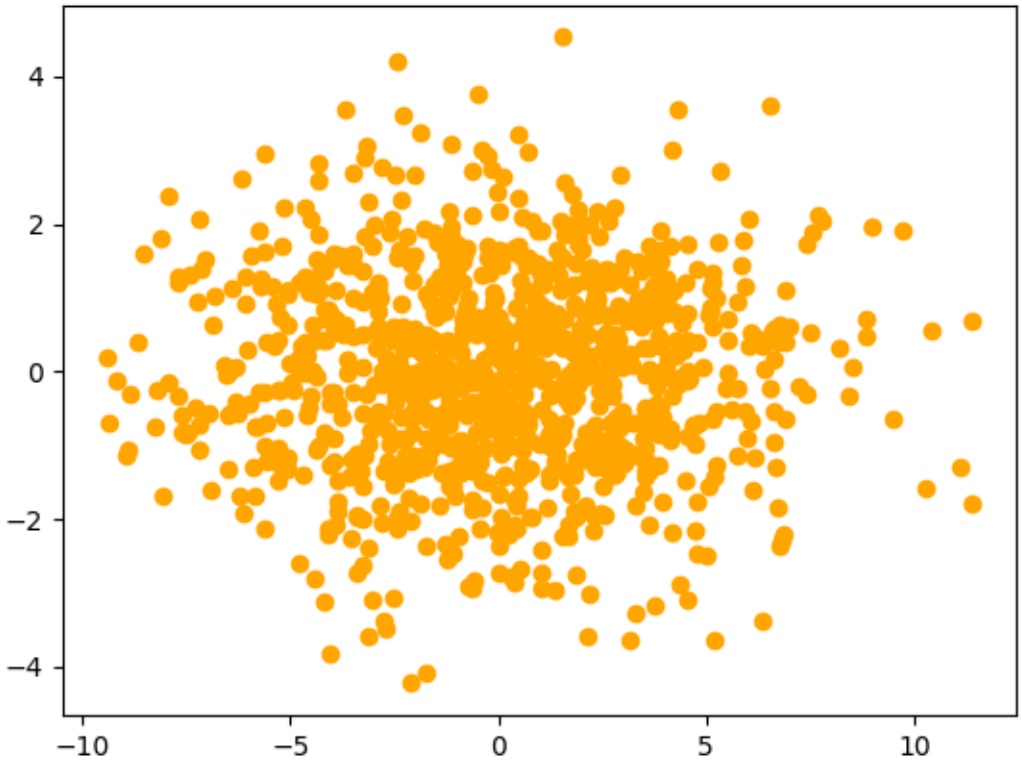
```
Drs covariance matrix =
    [[ 1.55653365e+01 -1.28722821e+01  4.13750509e+00 ...  7.37080401e-01
      -2.39302940e+01 -4.55958784e+00]
    [-1.28722821e+01  1.06451696e+01 -3.42164994e+00 ... -6.09553597e-01
       1.97899670e+01  3.77070556e+00]
    [ 4.13750509e+00 -3.42164994e+00  1.09981229e+00 ...  1.95927271e-01
      -6.36103903e+00 -1.21200835e+00]
    ...
   [ 7.37080401e-01 -6.09553597e-01  1.95927271e-01 ...  3.49036795e-02
      -1.13319430e+00 -2.15914563e-01]
   [-2.39302940e+01  1.97899670e+01 -6.36103903e+00 ... -1.13319430e+00
       3.67906578e+01  7.00995300e+00]
   [-4.55958784e+00  3.77070556e+00 -1.21200835e+00 ... -2.15914563e-01
       7.00995300e+00  1.33564997e+00]]
```

(c) (1 point) What is the total variance of the transformed data D_{RS}.
Drs total variance = 7.349058187383844
Question 2 Code:

```
R = np.array([[0.707, -0.707],
     [0.707, 0.707]])

S = np.array([[5, 0],
     [0, 2]])

RS = R * S


D_rs = np.empty([1000, 2])
for i in range(len(D)):
    D_rs[i, 0] = (D[i, 0] * RS[0,0]) + (D[i, 1] * RS[0,1])
    D_rs[i, 1] = (D[i, 0] * RS[1,0]) + (D[i, 1] * RS[1,1])

print("D_rs covariance matrix = ", np.cov(D_rs))
print("D_rs total variance = ", np.var(D_rs))

plt.scatter(D[:, 0], D[:, 1])
plt.scatter(D_rs[:, 0], D_rs[:, 1], color = "green")
plt.show()
```

3. (8 points) Use sklearn's PCA function to transform the data matrix D_{RS} from Question 2 to a 2-dimensional space where the coordinate axes are the principal components.

   (a) (4points) 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.



   (b) (2 points) What is the estimated covariance matrix of the PCA-transformed data?

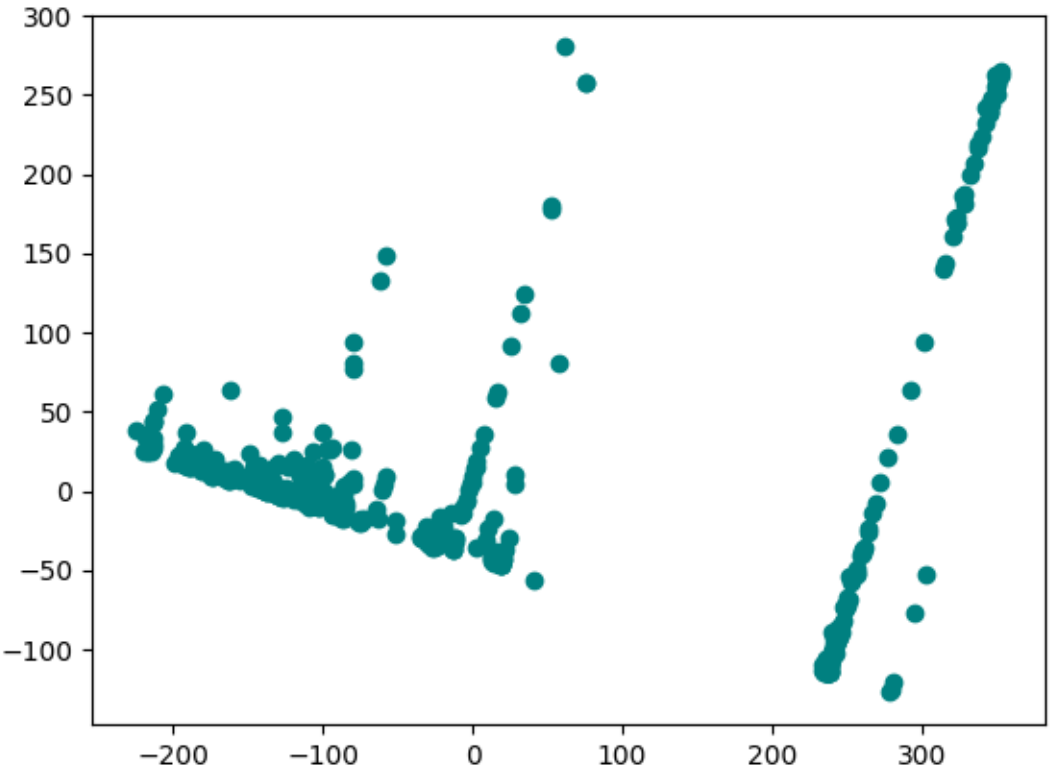
```
D_pca covariance matrix =  [[  9.7526634   -2.11908641  11.2666192  ...   1.58981398
  -16.70136789]
 [ -2.11908641   0.46044112  -2.44804303 ...  -0.3454393    0.96270246
    3.62892067]
 [ 11.2666192   -2.44804303  13.01559409 ...   1.83660894  -5.11843309
  -19.2940066 ]
 ...
 [  1.58981398  -0.3454393    1.83660894 ...   0.25916085  -0.72225362
   -2.72254533]
 [ -4.4306419    0.96270246  -5.11843309 ...  -0.72225362   2.01284375
    7.58744327]
 [-16.70136789   3.62892067 -19.2940066  ...  -2.72254533   7.58744327
   28.60097574]]
```

(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?
Fraction of variance captured by first principal component: 0.8654426212621087
Fraction of variance captured by first principal component: 0.1345573787378913
Question 3 Code:

```
pca = PCA(n_components = 2)
D_pca = pca.fit_transform(D_rs)

print("D_pca covariance matrix = ", np.cov(D_pca))
print("Fraction of variance captured by first principal component: "
      , pca.explained_variance_ratio_[0])
print("Fraction of variance captured by first principal component: "
      , pca.explained_variance_ratio_[1])


plt.scatter(D_pca[:, 0], D_pca[:, 1], color = "orange")
plt.show()
```

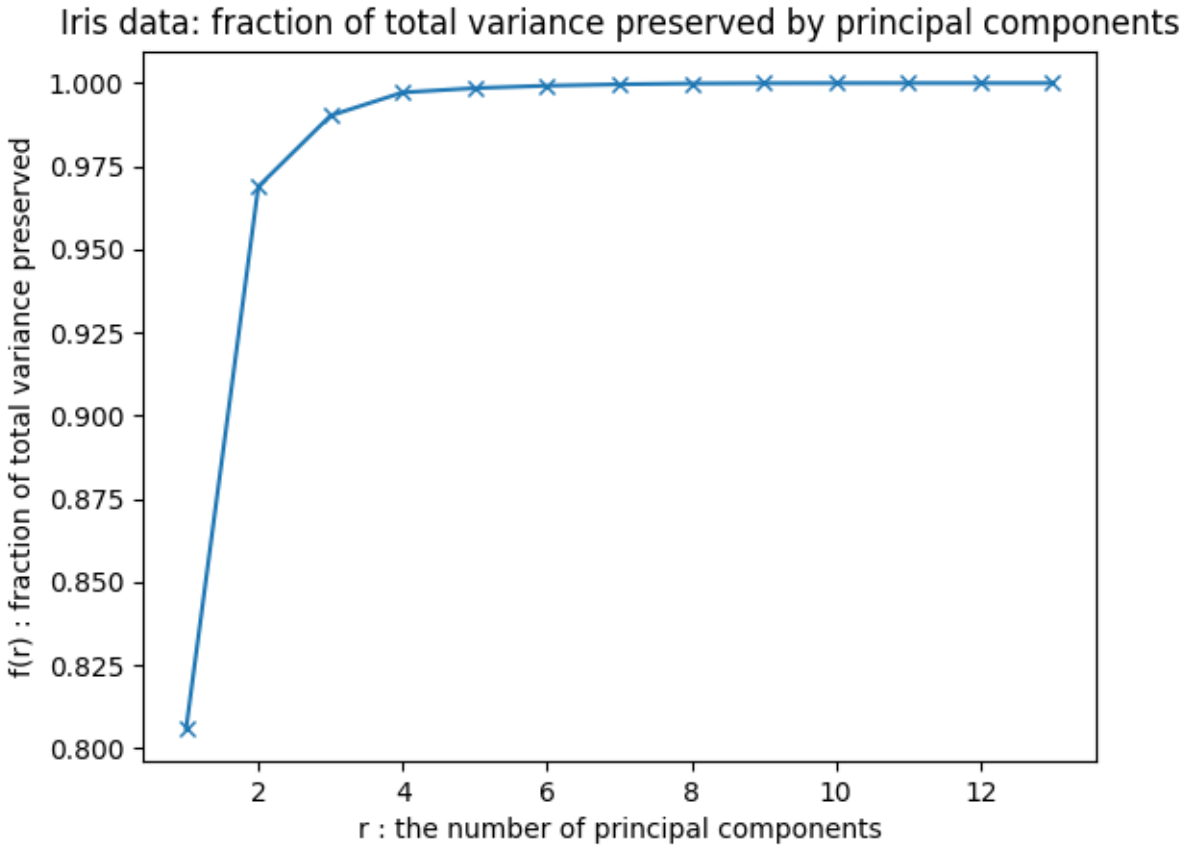
4. (18 points) Load the Boston data set into Python using sklearn's datasets package. Use sklearn's PCA function to reduce the dimensionality of the data to 2 dimensions.

   (a) (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.




```
boston = load_boston()['data']

multiMean = np.mean(boston, axis = 0)
bostonNormed = boston - multiMean
pca = PCA(n_components = 13)
Boston_pca = pca.fit_transform(bostonNormed)

plt.scatter(Boston_pca[:, 0], Boston_pca[:, 1], color = "teal")
plt.show()
```

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



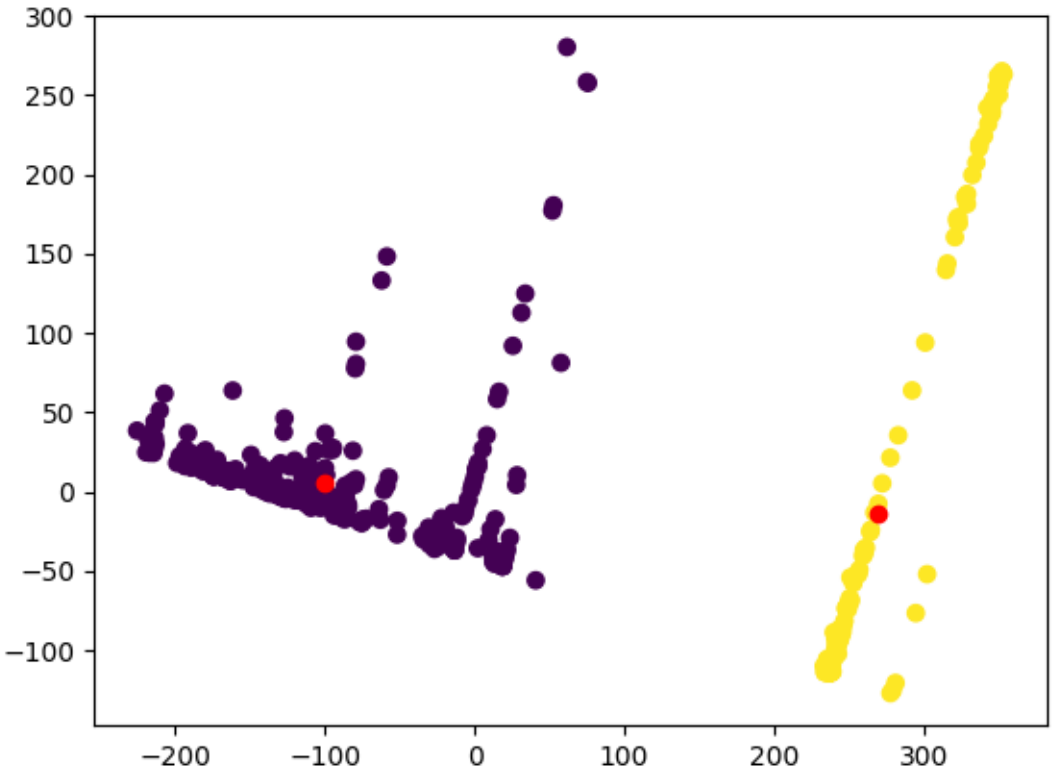

```
var_ratio = pca.explained_variance_ratio_
plt.plot(range(1,len(var_ratio)+1), np.cumsum(var_ratio), marker='x')
plt.title('Iris data: fraction of total variance preserved by principal components')
plt.xlabel('r : the number of principal components')
plt.ylabel('f(r) : fraction of total variance preserved')
plt.show()
```

(c) (2 points)

i. (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 should use at least two principal components if we want to capture at least 90 percent of the variance per above plot.

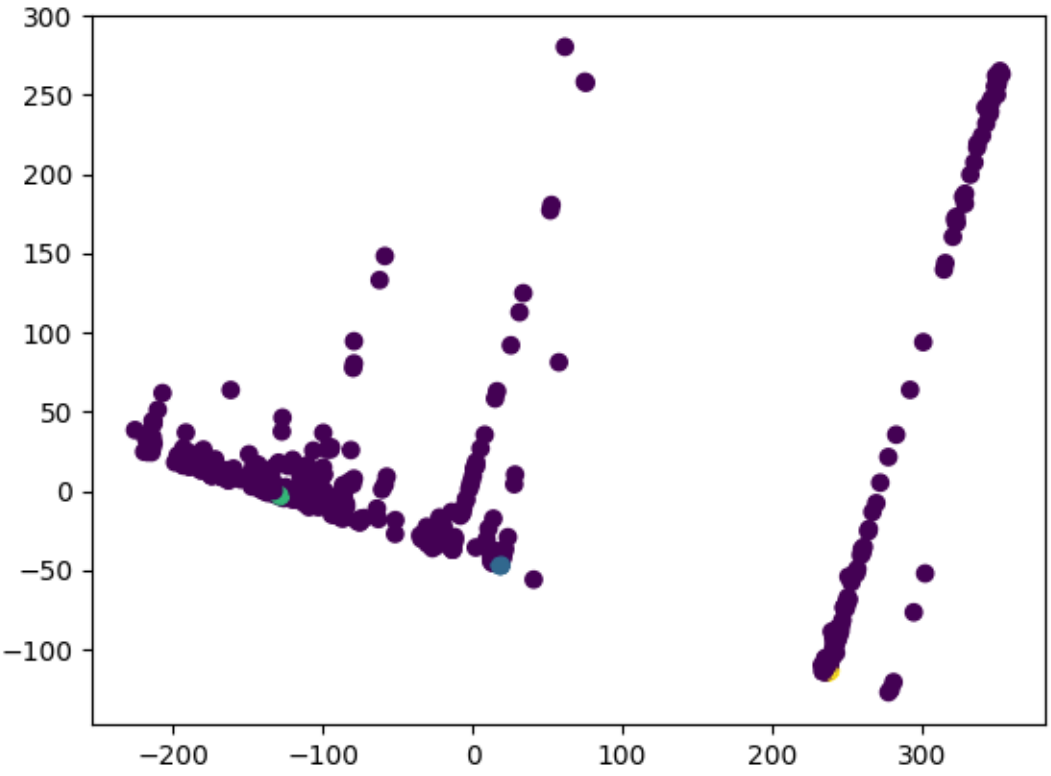
ii. (1 point) If we use two principal components of the normalized Boston data, how much (what fraction or percentage) of the total variance do we capture?
We capture 96.9 percent of the total variance by using two principal components.

(d) (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 4e). Plot the 2-dimensional data with colors corresponding to predicted cluster membership for each point. On the same plot, also plot the two means found by the k-means algorithm in a different color than the colors used for the data.




```
kmeans = KMeans(n_clusters=2, random_state=0).fit(Boston_pca)

centers = kmeans.cluster_centers_
labels = kmeans.fit_predict(Boston_pca)
plt.scatter(Boston_pca[:, 0], Boston_pca[:, 1], c = labels)
plt.scatter(centers[:, 0], centers[:, 1], c = 'red')
plt.show()
```

(e) (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). 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?




```
dbs = DBSCAN(eps=.6, min_samples=5).fit(Boston_pca)
labels = dbs.fit_predict(Boston_pca)

plt.scatter(Boston_pca[:, 0], Boston_pca[:, 1], c = labels)

plt.show()
```

This has three clusters: Cyan, Blue and Yellow

Acknowledgements: Homework problems adapted from assignments of Veronika Strnadova-Neeley.