final-project

May 12, 2022

River Kelly

Kyler Gappa

CSCI-347: Data Mining

Final Project

Part 1: Plan (20 points)

The problem that we are investigating is the association between developed liver disorders and the possible contributing factors that lead to such a diagnosis. It is no question that the consumption of alcoholic beverages increases the chances of developing a liver disorder, but the question still remains why some people are more at risk than others. In addition to alcohol consumption, other compounds found in the liver have been found to have confounding associations.

The data set that we have selected is the Liver Disorders Data Set (Forsyth). The data set includes a total of seven attributes. The first five attributes are numerical values pulled from the blood test of the patient. Since these values are thought to be sensitive to liver disease, they may show common traits so that early detection is more viable. The next variable is the number of drinks of alcohol per day. This could be a connecting factor for some of the test results. The final attribute is a categorical variable to help separate the data into test or training sets.

The data mining techniques that we will use to solve this problem include dimensionality reduction. There exists one categorical attribute within our data set that will need to be removed as it only contains information for if it was intended for a training dataset. We plan to use dimensionality reduction across all of the provided attributes in an attempt to identify which attributes have the greatest effect.

If we run out of time we plan to not explore the differences in the training and test categories to reduce the scope of the project from two separate datasets to a single dataset. If this were to happen we would totally ignore the categorical classification and run all testing over the entire dataset as a whole.

Part 2: Implement (30 points)

Setup Code

Import Libraries

[5]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from google.colab import drive
import numpy.linalg as LA
import os
```

Global (Const.) Variables

[6]:
```
# file path for the location of the data file
DATA_FILENAME = "/content/drive/My Drive/347-Data-Mining/Final-Project/data/bupa.
 ↪data"

# dictionary of columns for the data
DATA_COLUMN_NAMES = {
    'mcv': 'Mean Corpuscular Volume',
    'alkphos': 'Alkaline Phosphotase',
    'sgpt': 'Alamine Aminotransferase',
    'sgot': 'Aspartate Aminotransferase',
    'gmmagt': 'Gamma-Glutamyl Transpeptidase',
    'drinks': 'Number of half-pint equivalents of alcoholic beverages drunk per␣
 ↪day',
    # 'selector': 'Group' # field used to split data into two sets
}

DATA_COL_VALS = list(DATA_COLUMN_NAMES.values())

dot_colors = ['red', 'orange', 'blue', 'black', 'green']
```

Data Import Function

The following function is used read the data's file and parse the contents.

[7]:
```
# GetRawDataFromFile()
# returns a 2-dimensional array of the data
def getRawDataFromFile(file: str) -> list:

    # mount to google drive
    drive.mount('/content/drive')

    # check that file exists
```

```
    if not os.path.exists(file):
        # file does NOT exist, raise error
        raise RuntimeError(f'File "{file}" does not exist')

    # read file's lines
    lines = list()
    with open(file, 'r') as f: lines = f.readlines()

    # unmount google drive
    drive.flush_and_unmount()

    # parse the lines content and create the 2-dimensioal array
    data = list()
    # iterate through each line
    for line in lines:
        # clean the line string
        line = str(line).strip()

        # if line is empty, skip it
        if line is None or line == "" or len(line) < 1: continue

        # split the line (string) by seperator (",")
        line_data = line.split(',')

        # column 7 (index: 6) needs to be removed
        line_data.pop(6)

        # each value in the 'line_data' needs to be converted to a
        # numerical type.
        # All of the columns in the data are of type "int", except
        # for one, column[5], which is a float
        for data_item_index, data_item in enumerate(line_data):
            # is current column index 5?
            if data_item_index in [5]:
                # cast to type float
                data_item = float(data_item)
            else:
                # cast to type int
                data_item = int(data_item)
            # update the data_item value after type casting
            line_data[data_item_index] = data_item
        # append row to (raw) data array
        data.append(line_data)

    # return the raw data
    return data
```

Data Initialization

Get the `RAW_DATA` (2-dimensional array) from the file contents:

[8]:
```
# populate the raw data from the source file contents
RAW_DATA = getRawDataFromFile(DATA_FILENAME)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

Initialize `DataFrame`:

[9]:
```
# create a dataFrame from the RAW_DATA
df = pd.DataFrame(data=RAW_DATA, columns=DATA_COLUMN_NAMES.values())
```

Initialize `numpy.ndarray`:

[10]:
```
# create a numpy multi-dim. array from the RAW_DATA
D = np.ndarray(shape=(len(RAW_DATA), len(RAW_DATA[0])), dtype=float)
# populate the numpy data matrix (array)
for i, row in enumerate(RAW_DATA): D[i] = np.array(row)
```

Data Preview

The data is made up by 345 rows (instances) and 6 columns (attributes).

[11]:
```
D.shape
```

[11]:
```
(345, 6)
```

[12]:
```
D
```

[12]:
```
array([[85., 92., 45., 27., 31.,  0.],
       [85., 64., 59., 32., 23.,  0.],
       [86., 54., 33., 16., 54.,  0.],
       ...,
       [98., 77., 55., 35., 89., 15.],
       [91., 68., 27., 26., 14., 16.],
       [98., 99., 57., 45., 65., 20.]])
```

[13]:
```
df.describe()
```

[13]:

	Mean Corpuscular Volume	Alkaline Phosphotase	\
count	345.000000	345.000000	
mean	90.159420	69.869565	
std	4.448096	18.347670	
min	65.000000	23.000000	
25%	87.000000	57.000000	
50%	90.000000	67.000000	

```
75%                   93.000000             80.000000
max                  103.000000            138.000000

       Alamine Aminotransferase  Aspartate Aminotransferase  \
count                345.000000                  345.000000
mean                  30.405797                   24.643478
std                   19.512309                   10.064494
min                    4.000000                    5.000000
25%                   19.000000                   19.000000
50%                   26.000000                   23.000000
75%                   34.000000                   27.000000
max                  155.000000                   82.000000

       Gamma-Glutamyl Transpeptidase  \
count                      345.000000
mean                        38.284058
std                         39.254616
min                          5.000000
25%                         15.000000
50%                         25.000000
75%                         46.000000
max                        297.000000

       Number of half-pint equivalents of alcoholic beverages drunk per day
count                                          345.000000
mean                                             3.455072
std                                              3.337835
min                                              0.000000
25%                                              0.500000
50%                                              3.000000
75%                                              6.000000
max                                             20.000000
```

Estimated Covariance Matrix

[14]:
```
# get the covariave matrix
Sigma = np.cov(D.T, ddof=1)
Sigma
```

[14]:
```
array([[  19.7855578 ,    3.59934277,   12.81884058,    8.40583923,
          38.81795585,    4.6423576 ],
       [   3.59934277,  336.63700708,   27.28273509,   26.97080384,
          95.89180991,    6.17290192],
       [  12.81884058,   27.28273509,  380.73019885,  145.25846815,
         385.60532524,   13.47177283],
       [   8.40583923,   26.97080384,  145.25846815,  101.29403438,
         208.45331143,    9.39236603],
```

```
       [  38.81795585,   95.89180991,  385.60532524,  208.45331143,
        1540.92489046,   44.70902005],
       [   4.6423576 ,    6.17290192,   13.47177283,    9.39236603,
          44.70902005,   11.14114425]])
```

[15]:
```
# print the covariance matrix
frmt_str = '[ ' + ' '.join(["{:<7.2f}"] * Sigma.shape[1]) + ' ]'
for row in Sigma: print(frmt_str.format(*row))
```

```
[ 19.79   3.60    12.82   8.41    38.82   4.64    ]
[ 3.60    336.64  27.28   26.97   95.89   6.17    ]
[ 12.82   27.28   380.73  145.26  385.61  13.47   ]
[ 8.41    26.97   145.26  101.29  208.45  9.39    ]
[ 38.82   95.89   385.61  208.45  1540.92 44.71   ]
[ 4.64    6.17    13.47   9.39    44.71   11.14   ]
```

Eigenvalues

[16]:
```
evalues, evectors = LA.eig(Sigma)
```

[17]:
```
evalues
```

[17]:
```
array([1704.23455772,    8.51379717,   19.59323592,   37.76391713,
        290.92117087,  329.48615401])
```

[18]:
```
evectors
```

[18]:
```
array([[ 2.48915066e-02,  3.11252804e-01, -9.43765959e-01,
         1.08533543e-01,  4.94442234e-03,  1.98561179e-03],
       [ 7.49485205e-02,  6.73102408e-03,  1.32930005e-03,
        -4.14076557e-02, -3.81810281e-02,  9.95571826e-01],
       [ 2.92623798e-01, -1.30836279e-02, -3.21086702e-02,
        -3.49274184e-01,  8.89478140e-01, -2.31264856e-03],
       [ 1.50393856e-01,  5.10442210e-02,  1.28881103e-01,
         9.24576654e-01,  3.19083539e-01,  3.88528379e-02],
       [ 9.40591759e-01,  1.57223279e-02,  2.33472488e-02,
        -4.14320645e-02, -3.24855878e-01, -8.51287245e-02],
       [ 2.83421198e-02, -9.48710968e-01, -3.01857001e-01,
         8.91897708e-02,  8.68807215e-04,  8.42648015e-03]])
```

Sort eigenvalues (and corresponding vectors) in decending order.

[19]:
```
idx = evalues.argsort()[::-1]
evalues = evalues[idx]
evectors = evectors[:, idx]
```

[20]:
```
evalues
```

[20]:
```
array([1704.23455772,  329.48615401,  290.92117087,   37.76391713,
         19.59323592,    8.51379717])
```

[21]:
```
evectors
```

[21]:
```
array([[ 2.48915066e-02,  1.98561179e-03,  4.94442234e-03,
         1.08533543e-01, -9.43765959e-01,  3.11252804e-01],
       [ 7.49485205e-02,  9.95571826e-01, -3.81810281e-02,
        -4.14076557e-02,  1.32930005e-03,  6.73102408e-03],
       [ 2.92623798e-01, -2.31264856e-03,  8.89478140e-01,
        -3.49274184e-01, -3.21086702e-02, -1.30836279e-02],
       [ 1.50393856e-01,  3.88528379e-02,  3.19083539e-01,
         9.24576654e-01,  1.28881103e-01,  5.10442210e-02],
       [ 9.40591759e-01, -8.51287245e-02, -3.24855878e-01,
        -4.14320645e-02,  2.33472488e-02,  1.57223279e-02],
       [ 2.83421198e-02,  8.42648015e-03,  8.68807215e-04,
         8.91897708e-02, -3.01857001e-01, -9.48710968e-01]])
```

Total Variance

[22]:
```
total_var = sum(np.diag(Sigma))
total_var
```

[22]:
```
2390.512832827772
```

Mean Centering

First, we must find the multi-dimensional mean:

[23]:
```
multi_d_mean = np.mean(D, axis=0)
multi_d_mean
```

[23]:
```
array([90.15942029, 69.86956522, 30.4057971 , 24.64347826, 38.28405797,
       3.45507246])
```

Now we can center our data from the multi-dimensional mean:

[24]:
```
centered_data = D - multi_d_mean
centered_data
```

[24]:
```
array([[ -5.15942029,  22.13043478,  14.5942029 ,   2.35652174,
         -7.28405797,  -3.45507246],
       [ -5.15942029,  -5.86956522,  28.5942029 ,   7.35652174,
        -15.28405797,  -3.45507246],
       [ -4.15942029, -15.86956522,   2.5942029 ,  -8.64347826,
         15.71594203,  -3.45507246],
       ...,
       [  7.84057971,   7.13043478,  24.5942029 ,  10.35652174,
```

```
  50.71594203,  11.54492754],
[  0.84057971,  -1.86956522,  -3.4057971 ,   1.35652174,
 -24.28405797,  12.54492754],
[  7.84057971,  29.13043478,  26.5942029 ,  20.35652174,
  26.71594203,  16.54492754]])
```

Principle Component Analysis (PCA)

[25]:
```
pca = PCA(n_components=2)
```

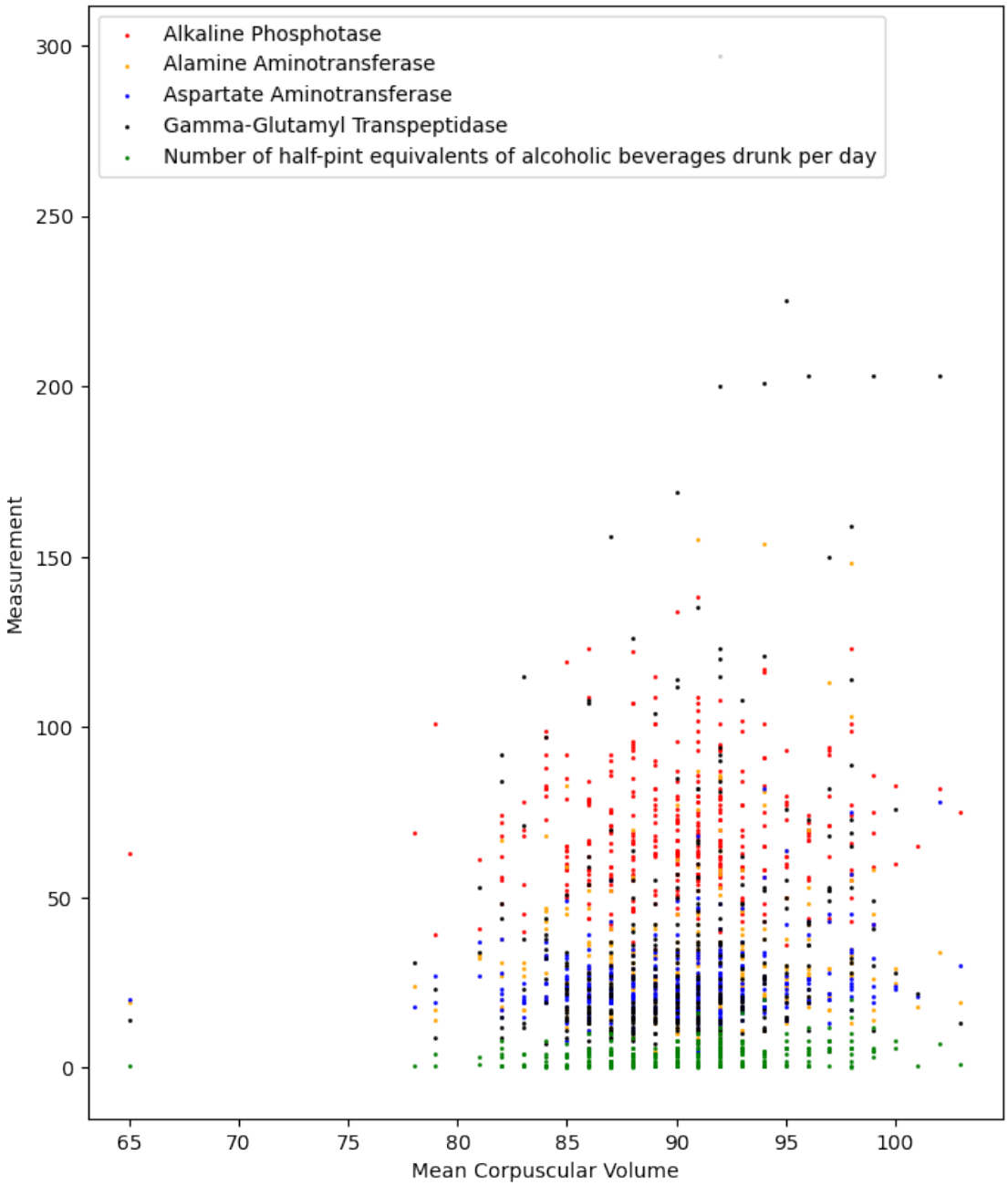
[26]:
```
pca_transformed_D = pca.fit_transform(D)
pca_transformed_centered_D = pca.fit_transform(centered_data)
```

Data Visualization

Raw Data

Blood tests which are thought to be sensitive to liver disorder that might arise from excessive alcohol consumption.

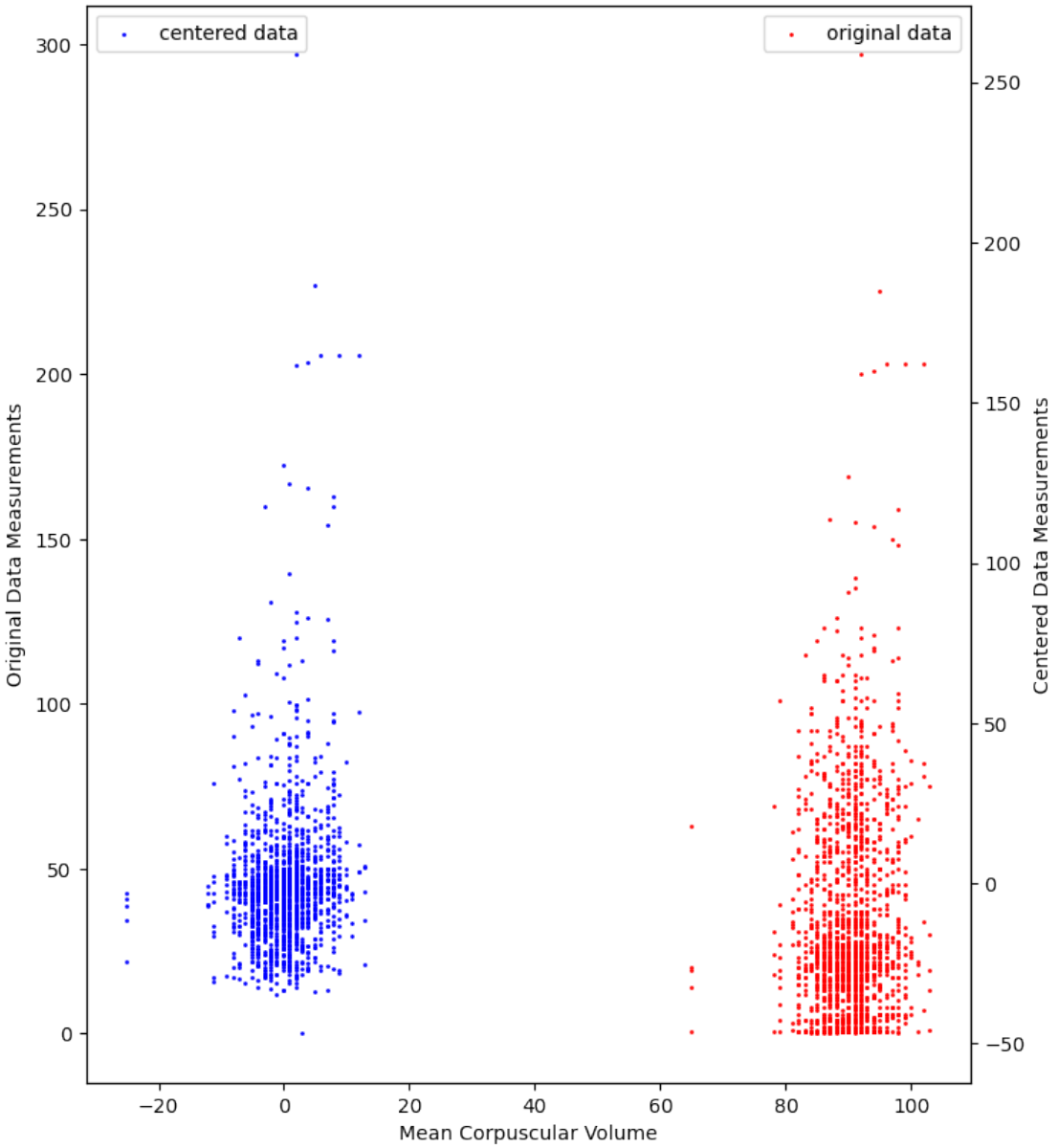
[27]:
```
plt.figure(figsize=(8, 10), dpi=100)
for i in range(1, 6):
    plt.scatter(D[:,0], D[:,i], s=1, c=dot_colors[i-1], label=DATA_COL_VALS[i])
plt.ylabel('Measurement')
plt.xlabel(DATA_COL_VALS[0])
plt.legend(loc='upper left')
plt.show()
```



Original Data v.s. Mean Centered Data

[28]:
```
fig = plt.figure(figsize=(8, 10), dpi=100)
ax1 = fig.add_subplot(111)
ax2 = ax1.twinx()
for i in range(1, 6):
    ax1.scatter(D[:,0], D[:,i], s=1, c='red')
```

```
for i in range(1, 6):
    ax2.scatter(centered_data[:,0], centered_data[:,i], s=1, c='blue')
ax1.legend(['original data'], loc='upper right')
ax2.legend(['centered data'], loc='upper left')
ax1.set_ylabel('Original Data Measurements')
ax2.set_ylabel('Centered Data Measurements')
ax1.set_xlabel(DATA_COL_VALS[0])
plt.show()
```



Centered Data v.s. Projected Data

Let's project our data on to the largest eigenvector.

[29]:
```
Sigma_centered = np.cov(pca_transformed_centered_D.T, ddof=1)

evalues_centered, evectors_centered = LA.eig(Sigma_centered)

# sort
idx = evalues_centered.argsort()[::-1]
evalues_centered = evalues_centered[idx]
evectors_centered = evectors_centered[:, idx]

coords_along_eig0 = evectors_centered[:,0].T.dot(pca_transformed_centered_D.T)
coords_along_eig1 = evectors_centered[:,1].T.dot(pca_transformed_centered_D.T)
coords_along_eig0
```

[29]:
```
array([-7.94013907e-01, -5.56860410e+00,  1.28506263e+01, -6.61028704e-01,
       -3.16517168e+01, -2.72797043e+01, -3.24661762e+01, -3.08199894e+01,
       -3.38109526e+01, -3.45647505e+01, -2.85428289e+01, -2.78498460e+01,
       -1.42506455e+01, -2.11624988e+01, -2.69263222e+01, -2.06214855e+01,
       -3.03984727e+01, -2.54608889e+01,  1.56401760e+01, -2.66157360e+01,
       -1.93977566e+01, -2.00776630e+01, -3.39468473e+01, -2.26958953e+01,
        6.18625502e+01, -1.52080423e+01, -2.30350924e+01, -3.06242149e+01,
       -2.14670900e+01, -2.39611948e+01, -2.15265037e+01,  2.39450873e+00,
       -8.74377620e+00,  2.18298389e+00, -2.94751566e+01,  8.41954814e+01,
       -1.06056017e+01, -3.58689708e+01, -8.24794406e+00, -3.46749835e+00,
        1.96572444e+01,  4.11234216e+00, -2.51380221e+01, -2.53193786e+01,
       -2.28387419e+01, -2.84698060e+01, -3.37818461e+01,  9.20970930e+00,
       -2.76135229e+01, -3.12557509e+00, -2.43480207e+01, -2.85873056e+01,
        5.91795101e+01, -3.11723763e+00, -1.28947144e+01, -2.39898569e+01,
       -2.07666550e+01, -2.32826200e+01, -3.38703651e+00, -1.60093160e+01,
       -1.03990614e+01, -2.52581126e+01, -3.51892997e+01, -3.19709530e+01,
       -1.10594350e+01, -1.96900438e+01, -3.17387692e+01, -1.73743289e+01,
       -2.56794384e+01, -2.81932345e+01,  1.63854503e+01, -2.85360694e+00,
       -2.93484245e+01, -1.46234448e+01, -2.58873164e+01, -7.29083906e-01,
        1.16301672e+02, -9.08084068e+00, -2.83051967e+01, -8.04630172e+00,
        2.83832954e+01,  1.03569424e+01, -1.41477256e+01, -1.61410305e+01,
        2.42846609e+02, -1.61410305e+01, -9.57840152e+00, -3.17739719e+01,
       -2.81040173e+01, -3.15990006e+01, -1.33056278e+01, -3.58649769e+01,
       -2.27294490e+00, -2.63260741e+01, -1.36310763e+01, -1.53614485e+01,
        1.52608902e+01,  6.00376631e+01, -4.84346965e+00, -1.41032602e+01,
       -1.57310722e+01,  3.41650803e+01, -2.92373608e+01, -2.65888457e+01,
       -2.34268121e+01, -2.29878703e+01, -8.13561148e+00, -1.68010763e+01,
       -3.66312398e+01,  1.26547626e+01, -2.50423768e+01, -1.60626511e+01,
       -2.37080007e+01, -2.11195252e+01,  1.29635072e+02, -3.59170128e+01,
       -2.24706213e+01, -2.81648875e+01, -1.66571795e+01, -2.76964173e+01,
        9.84917291e+00, -1.58736582e+01, -2.35225047e+01, -1.83380989e+01,
```

```
-5.21300315e+00, -2.84349426e+01,  7.01217024e+00,  3.98415774e+01,
-2.96942425e+01, -1.99339260e+01, -9.22284434e+00, -2.17677681e+01,
 7.21982565e+01,  1.30584793e+02, -3.09289297e+01, -2.87036818e+01,
-1.88240683e+01, -1.91911334e+01,  6.75571679e+01, -3.09855758e+01,
-2.20878390e+01, -1.19069955e+01, -2.37004572e+01, -1.23857098e+00,
-2.98914338e+01,  1.26284222e+01,  9.86913782e+00,  4.75188874e+01,
-2.09618560e+01, -2.37004572e+01,  6.20713708e+01, -1.27406522e+00,
-2.18859075e+01, -3.11195147e+01,  3.43941317e+01,  2.82418243e+01,
 4.77868760e+01,  1.46139275e+01,  1.45090132e+01, -2.26821924e+01,
 2.64758524e+01, -2.49425819e+01, -8.70750716e+00,  3.81068131e+00,
-2.84245991e+01, -1.04233978e+01,  5.65232182e+01,  8.83734968e+01,
 3.49742559e+01,  1.24759791e+01, -9.48512403e+00,  3.41344077e+01,
-1.97299031e+01, -3.31627144e+01,  8.69212714e+01,  1.24759791e+01,
 2.46257040e+01, -2.80777293e+01,  1.73655039e+02,  1.07905400e+01,
 3.17153878e+01,  7.96647426e+01,  1.26425097e+01, -1.43188504e+01,
 3.36506764e+01,  2.84408340e+01,  6.34201428e+01,  1.01954878e+01,
 7.09081255e+01,  1.70810128e+02, -2.47632547e+01, -2.90799978e+01,
-1.72682324e+00, -2.20002213e+01, -3.74839664e+01, -2.59067183e+01,
-2.90954503e+01, -2.15012307e+01, -1.74055503e+01, -2.91022786e+01,
-1.95560567e+01, -2.86208677e+01,  2.22451181e+01, -1.99099385e+01,
 9.38126360e+01, -6.06023867e+00, -2.76379176e+01, -2.02262355e+01,
-1.64169712e+01, -3.34981419e+01, -1.64642682e+01, -1.01765438e+01,
-7.46453247e+00, -2.97241294e+01, -2.85930086e+01, -2.02102714e+01,
-3.31876259e+01,  1.85884839e+01, -2.18144432e+01,  9.26434637e+00,
-2.61755367e+00, -1.79194272e+01, -2.81464522e+01, -2.81022157e+01,
-2.99045277e+01, -2.57889930e+01,  2.00286155e+00,  2.97032682e+01,
 1.18884964e+01,  2.03997540e+01, -1.27592347e+01, -1.57567593e+01,
 1.55949575e+02, -2.86218448e+00,  2.57490567e+01, -8.15251311e+00,
-1.96737203e+01, -9.53444531e+00, -3.08101738e+01, -1.99156007e+01,
-2.23778522e+01, -3.07255197e+01, -2.56423174e+01, -5.37732266e+00,
-2.33751907e+01, -2.57318869e+01, -2.25193348e+01, -3.04075154e+01,
-2.80691506e+01,  4.75260521e+01, -2.84727865e+00,  7.18450897e+01,
-2.55019474e+01,  1.22309054e+01,  5.23915762e+00, -1.52910340e+01,
-2.44874860e+01, -1.74928034e+01, -2.19798612e+01, -7.61009312e+00,
 3.55174703e+01, -1.50206801e+01, -1.01976138e+01, -2.09646309e+01,
 3.11772659e+01, -1.07715560e+01, -3.18157857e+01, -8.98664570e+00,
-2.68006294e+00, -2.32313106e+01, -2.97339128e+01, -1.77252253e+01,
-3.13539472e+01, -2.58348536e+01, -1.58880656e+01, -5.20244477e+00,
 4.24192406e+01,  2.99829671e+01, -6.22141837e+00,  1.16201089e+01,
-1.31915122e+01, -2.44249529e+01, -3.13146674e+01, -1.93006472e+01,
-1.86349889e+01,  3.48949015e+01, -2.53143407e+01, -1.17550748e+01,
 6.43976833e+00,  1.70666800e+00, -2.98611344e+00, -2.79760520e+01,
-2.37412567e+01,  6.55178864e+01,  2.38276831e+01,  1.06442486e+01,
-8.18873375e+00,  1.69826668e+01, -2.19834009e+01,  1.20691815e+02,
-1.28001380e+01, -3.31428015e+01, -3.07920310e+01,  2.90570255e+00,
 5.92353668e+00, -8.00506849e+00,  7.46488945e+01, -1.18166635e+01,
-2.90473276e+01, -2.93237018e+00,  5.66614903e+01,  4.96158678e+01,
```

```
 5.93276227e+00, -2.10093392e+01, -2.28731489e+01,  1.67116629e+02,
 9.73040788e+01, -2.20878390e+01,  4.01985705e+00,  7.72639940e+00,
-1.13847795e+01, -1.20970644e+00,  1.65311059e+02, -3.07436469e+01,
-2.50254305e+01, -1.76704030e-01,  6.07976538e-01, -1.27709048e+01,
-3.34606936e+00,  3.10228175e-01,  1.73116035e+02,  1.02788343e+01,
-9.99387384e+00,  9.32390209e+01,  1.81740850e+01, -1.60813582e+00,
-1.33034138e+01,  3.48377442e+00, -2.90276768e+01,  2.38699141e+01,
 2.01533164e+00,  1.64625157e+02,  5.75141905e+01, -2.33976376e+01,
 3.88197525e+01])
```

Observe that now, we now have a 1 dimensional representation of our data that captures about 73% of the total variance of the centered data set.

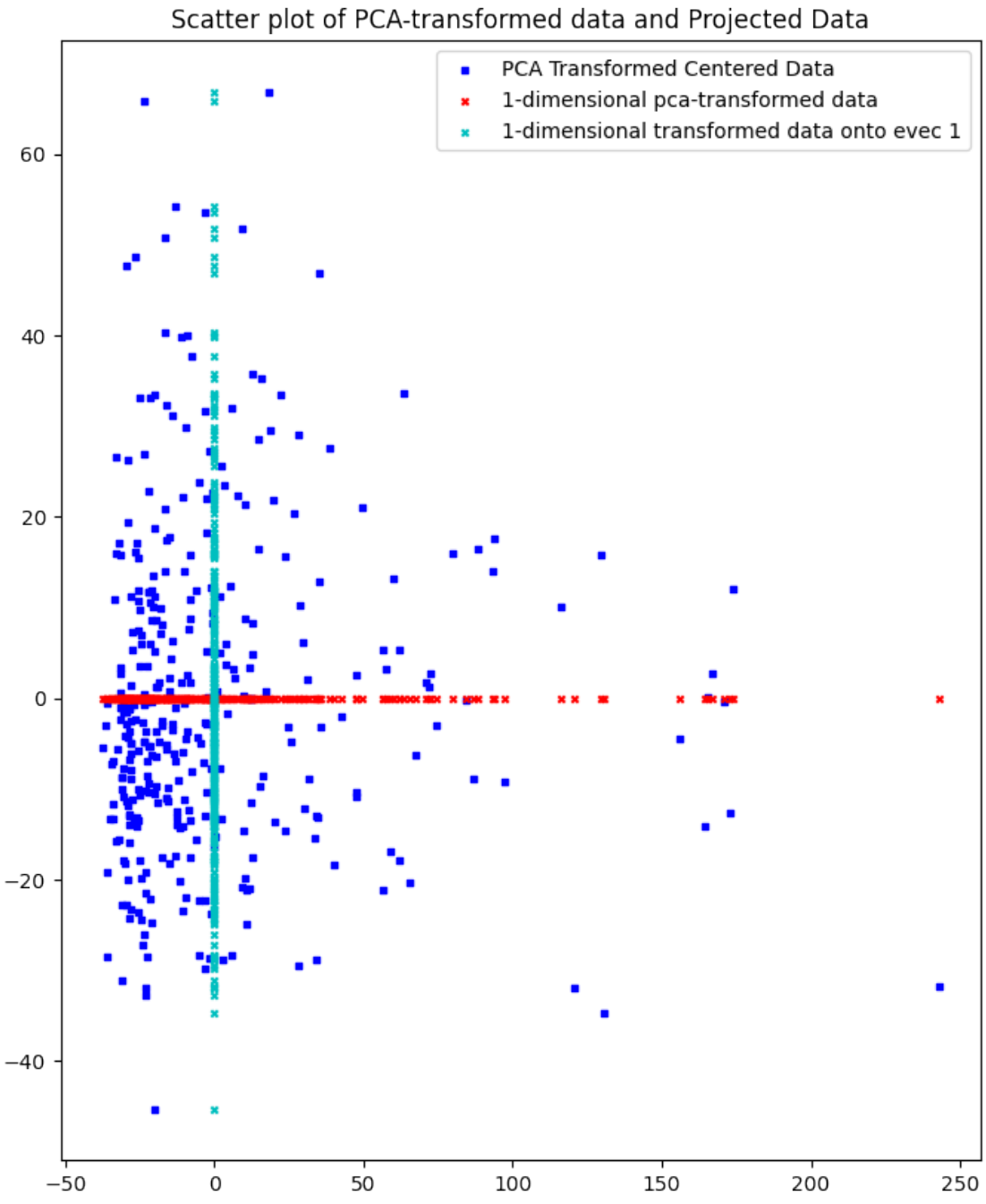
[30]:
```
evalues_centered[0]/sum(np.diag(Sigma_centered))
```

[30]:
```
0.8379884946288922
```

[31]:
```
projected_along_eig0 = np.zeros((D.shape[0],2))
for i in range(len(coords_along_eig0)):
    projected_along_eig0[i,:] = coords_along_eig0[i]*evectors_centered[:,0]
projected_along_eig1 = np.zeros((D.shape[0],2))
for i in range(len(coords_along_eig1)):
    projected_along_eig1[i,:] = coords_along_eig1[i]*evectors_centered[:,1]

pjdev0 = projected_along_eig0
pjdev1 = projected_along_eig1

fig = plt.figure(figsize=(8, 10), dpi=100)
ax = fig.add_subplot(111)
ax.scatter(pca_transformed_centered_D[:,0], pca_transformed_centered_D[:,1],␣
 ↪s=10, c='b', marker='s', label='PCA Transformed Centered Data')
ax.scatter(pjdev0[:,0], pjdev0[:,1], s=10, c='r', marker='x',␣
 ↪label='1-dimensional pca-transformed data')
ax.scatter(pjdev1[:,0], pjdev1[:,1], s=10, c='c', marker='x',␣
 ↪label='1-dimensional transformed data onto evec 1')
plt.legend(loc='upper right')
plt.title('Scatter plot of PCA-transformed data and Projected Data')
plt.show()
```



Part 3: Report (40 points)

Problem Statement

We where hoping to help identify key characteristics that can help show potential signs of liver disorders.

Data

This data set had 345 intstances and 7 attributes with no missing values.

All of the attributes in our data were numerical.

Preprocessing Techniques

To preprocess our data, we used a few techiques. We mean centered our data so that our data results would be standardized between the techiques that require mean centering and those that do not.

Data Mining Techniques

We used dimensionality reduction to identify which of the attributes has the greatest effect on PCA. We used PCA as it very quickly can give us a good estimation on which attributes hold the majority of the variance. It can also show us how greatly an attribute effects the position of a data point when a dimension is reduced.

Analysis

Data was taken from *https://archive.ics.uci.edu/ml/datasets/Liver+Disorders*

Proportion of Total Variance

In the direction of the largest eigenvalues, we capture 71% of the total variance. This shows us that while there is a large amount of data in the major components, the other components may still contain valuable information.

[32]:
```
evalues[0]/total_var
```

[32]:
```
0.7129158791006928
```

Missing Values

There were no missing values from the data.

[33]:
```
df.isnull().sum()
```

[33]:
```
Mean Corpuscular Volume                                    0
Alkaline Phosphotase                                       0
Alamine Aminotransferase                                   0
Aspartate Aminotransferase                                 0
Gamma-Glutamyl Transpeptidase                              0
Number of half-pint equivalents of alcoholic beverages drunk per day    0
dtype: int64
```

Results

Time Constraints

We did not explore everything that we hoped to be able to when we started the project. We where not able to run the same techiques on the same dataset by separating the datasets into a test base and a real data. We where also not able to run the k-means algorithm.

Video Link

https://youtu.be/COPckqaGQb8

References

Forsyth, Richard S. "Liver Disorders Data Set." *UCI Machine Learning Repository*, 15 May 1990, *https://archive.ics.uci.edu/ml/datasets/Liver+Disorders*. Accessed 14 April 2022.