Assignment 1

Gengcong Yan - 1009903
CS-E4650 Methods of Data mining

October 3, 2021

1 Task 1

1.1 a)

I calculate all pairwise correlations between features, excluding only the rat id. The functions below I implemented are for calculation.

```
def check_strong(x):
    if np.abs(x)>0.4:
        return x
    else:
        return ""

def get_correlation(data):
    p_corr=data.corr(method='pearson')
    p_corr=p_corr.round(2)
    p_corr
    corr_strong=p_corr.applymap(check_strong)
    return corr_strong
```

The correlation result is shown in fig 1 after deleting the values less than 0.40. We can see some strong correlations in the form of features pairs below:

1. femsate-(gender), kmethod-(place)
2. day-(kmethod,place,gonind), year-(kmethod,place,BMI)
3. weight-(blength,ADWBind,gonind,BMI),gonfatind-(appind,kmethod,place,gonind)

1.2 b)

After removing outlier rats, rat2, rat53, rat120, and rat434, and calculate correlations again.we can see liveind starts to show strong correlations with ADWBind and gonind with the value 0.46 and -0.48. Heartind also starts to show strong correlations with many features

	day	weight	gender	liverind	heartind	appind	femstate	gonfatind	batind	sulcer	kmethod	tailind	blength	place	year	ADWBind	gonind	BMI
day	1										0.45			0.7			0.44	
weight		1											0.88			-0.41	0.53	0.88
gender			1				-0.89											
liverind				1														
heartind					1													
appind						1		0.43										
femstate			-0.89				1											
gonfatind						0.43		1			0.49			0.57			0.68	
batind									1									
sulcer										1								
kmethod	0.45							0.49			1			0.77	0.54		0.52	
tailind												1	-0.42					
blength		0.88										-0.42	1				0.44	0.6
place	0.7							0.57			0.77			1	0.46		0.66	
year											0.54			0.46	1			0.43
ADWBind		-0.41														1	-0.53	
gonind	0.44	0.53						0.68			0.52		0.44	0.66		-0.53	1	0.5
BMI		0.88											0.6		0.43		0.5	1

Figure 1: Strong Correlations Bewteen Features.

like blength and ADWBind. Because some of their values in liverind or heartind about our deleted outliers are apparently unreasonable. they either took up too much or too litter percent of whole body weights, which is impossible.

1.3 c)

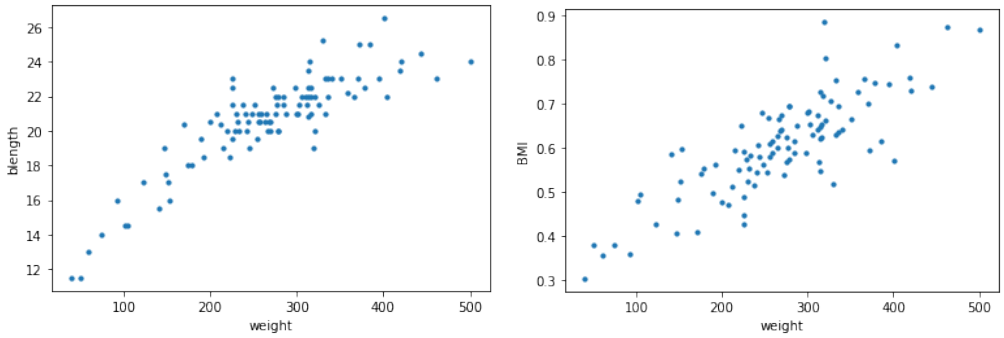
```
freezer_index=data_dropb.loc[data_dropb["day"]==400].index
data_dropb.loc[freezer_index,"day"]=0
data_dropb.loc[freezer_index,"year"]=-1
corr3=get_correlation(data_dropb)
corr3

data_dropb=data_dropb.drop(freezer_index)
corr4=get_correlation(data_dropb)
corr4
```

We change the special codes for the freezer rats: day=0 and year=-1, remove them last to see correlations differences. the correlation of day and years with many other features are vanished and extremely negative. I think day and year don't have strong correlations with other features, those time related feature can't maintain obvious connection between the actual physiological characteristics.

1.4 e)

```
data_dropb["femstate"]=data_dropb["femstate"]+10
data_dropb["place"]=data_dropb["place"]+5
corr5=get_correlation(data_dropb)
corr5
```



(a) Weight-Blength (b) Weight-BMI

Figure 2: Features Correlation Plots

We change special codes of categorical features femstate, kmethod and place by add numbers to the original values. And we can see the correlations related to them doesn't change at all. Because they are just categorical features in data, which mean the codes represent some certain category without meaning any Mathematical values. So you can change them to any code you like as long as it's the same number in one category.

1.5 f)

From the observation in correlations matrix, I think ***weight, blength,gonind,BMI, gonfatind*** are reliable features and shows strong linear trend to each other. When one of the metrics increases or decreases, the others will increase or decrease as well. Because weight and height are the most basic characteristics, all other indicators based on them show a strong correlation with them. From the scatter plot in Fig.2, we can observe the linear trend between different features like Weight, Blength and BMI.

2 Task 2 (Extra)

First,we will get the clean dataset from Task 1 after removing the listed outliers but keeping only wild rats (place 1-3).Then we convert day feature into circular form using $\theta = day/365 * 2\pi$.Then we can calculate the Pearson correlation coefficient as the basis of the R^2 correlation shown in Fig.3. With the function written in Appendix B in Task document, we can calculate the R^2 of (day,gonfatind) and (day,batind),which are $R_g^2 = 0.017, R_g = 0.132$ and $R_b^2 = 0.232, R_b = 0.486$.

```

# convert and get related data features
data_task2["day_c"]= data_task2["day"]/365*2*np.pi
data_task2["day_sin"]= np.sin(data_task2["day_c"])
data_task2["day_cos"]= np.cos(data_task2["day_c"])
data_task2=data_task2[["day","day_c","day_sin","day_cos","
    gonfatind","batind"]]

# Compute correlations
corr_task2=data_task2.corr(method='pearson').round(3)


# numbers are from correlation matrix

# day--gonfatind
R2_g=((-0.087)**2+(-0.116)**2-2*(-0.087*-0.116*0.221))
    /(1-0.221**2)
print(R2_g)
print(np.sqrt(R2_g))

#day --batind
R2_b=((0.341)**2+(0.407)**2-2*(0.341*0.407*0.221))/(1-0.221**2)
print(R2_b)
print(np.sqrt(R2_b))
```

	day	day_c	day_sin	day_cos	gonfatind	batind
day	1.000	1.000	-0.826	-0.468	0.124	-0.441
day_c	1.000	1.000	-0.826	-0.468	0.124	-0.441
day_sin	-0.826	-0.826	1.000	0.221	-0.116	0.407
day_cos	-0.468	-0.468	0.221	1.000	-0.087	0.341
gonfatind	0.124	0.124	-0.116	-0.087	1.000	-0.058
batind	-0.441	-0.441	0.407	0.341	-0.058	1.000

Figure 3: Day Related Matrix

From the indices I computed above, when turning day feature into circular form,we know there is no strong correlation between the time of a year and gofatind, but it shows strong correlation between the time of a year and batind.

3 Task 3

The code of principal component analysis:

```
def zeroMean(dataMat):
    meanVal=np.mean(dataMat, axis=0)
    newData=dataMat-meanVal
    return newData,meanVal

def pca(dataMat,n):
    newData,meanVal=zeroMean(dataMat)
    covMat=np.cov(newData, rowvar=0)

    eigvals,eigVects=np.linalg.eig(np.mat(covMat))
    eigValIndice=np.argsort(eigvals)
    # print(eigValIndice)
    n_eigvalIndice=eigValIndice[ -1:-(n+1):-1] # pick n values
       from last to start
    # print(n_eigvalIndice)

    # choose the smallest one
    # n_eigvalIndice=1  # 1 is the index of the smallest
       eigenvalue

    n_eigvect=eigVects[ :,n_eigvalIndice]
    # print(n_eigvect)
    lowDDataMat=newData*n_eigvect
    reconMat=( lowDDataMat*n_eigvect.T)+meanVal
    return eigvals,eigVects,lowDDataMat ,reconMat
```

3.1 a)

The result of eigenvalue decomposition of the corresponding sample covariance matrix is [1.5 , 0.16666667]

3.2 b

1D representation under the**largest** eigenvalue is [[**-1.06066017**],[**-1.06066017**],[**1.06066017**],[**1.06066017**]]. And 1D representation under the **smallest** eigenvalue is [[**-0.35355339**],[**0.35355339**],[**-0.35355339**],[**0.35355339**]]

3.3 c

The code of disance computation

```python
def get_dis_pairs(M):
    lens=M.shape[0]
    result=np.zeros((lens,lens))
    for i in range(lens):
        j=i+1
        while(j<lens):
            result[i][j]=np.linalg.norm(M[i]-M[j])
            j=j+1
    return result
```

After euclidean distance computation, we can see the result from Fig.4. The distance between points in original matrix and reconstructed matrix after PCA are the same. But, the distance between points after PCA changed due to the dimension decreasing.

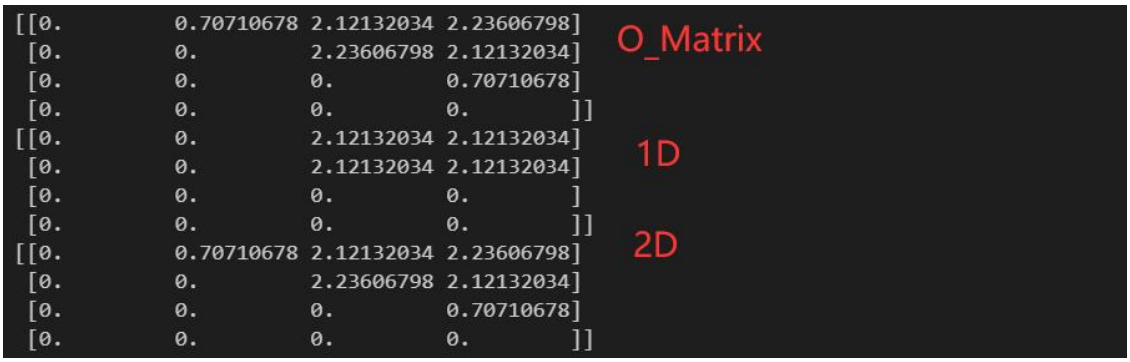
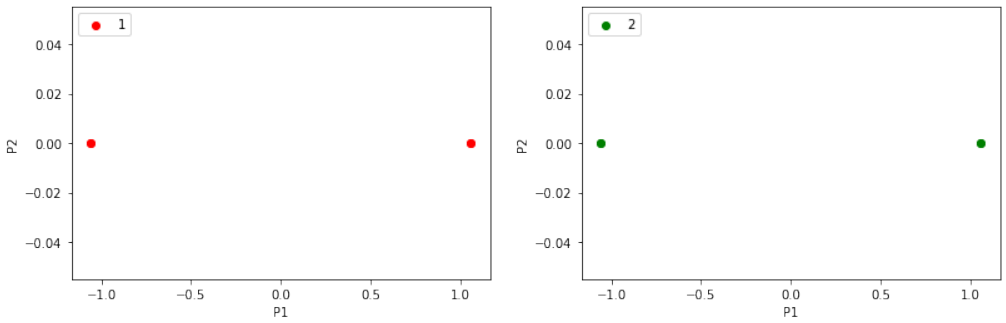


Figure 4: Euclidean distance

3.4 d)

In new dataset. the eigenvalue decomposition of the corresponding sample covariance matrix is [1.5, 0.16666667], the same as in the old dataset. when converting two datasets into 1-D form, points in their own graph are lies on the same places in Fig.5 they represent same information after principal component analysis. These two datasets represent different information on the original data space, but show the same data characteristics in one dimensional space after PCA process.



(a) Old dataset (b) New dataset

Figure 5: Points in 2 datasets

4 Task 4

4.1 a)

```
#scale
cows["age_scaler"]=(cows["age"]-cows["age"].min())/(cows["age"].max()-cows["age"].min())
cows["milk_scaler"]=(cows["milk"]-cows["milk"].min())/(cows["milk"].max()-cows["milk"].min())
cows

ndata=np.array(cows[["age_scaler","milk_scaler"]])

# Compute numerical similarity
def get_numsim_M(ndata):
    lens=len(ndata)

    SM=np.zeros((lens,lens))

    for i in range(lens):
        j=i+1
        while j <lens:
            tep=(ndata[i][0]-ndata[j][0])**2+(ndata[i][1]-ndata[j][1])**2
            tep=np.sqrt(tep)
            SM[i][j]=SM[j][i]=tep
            j=j+1
    return SM.round(4)

re=get_numsim_M(ndata)
re
```

The euclidean distances between cows is shown in Fig.6

```
array([[0.    , 0.2857, 0.52  , 0.3627, 0.9371, 1.2289],
       [0.2857, 0.    , 0.52  , 0.5429, 1.0976, 1.4142],
       [0.52  , 0.52  , 0.    , 0.3308, 0.6615, 0.9923],
       [0.3627, 0.5429, 0.3308, 0.    , 0.5759, 0.8781],
       [0.9371, 1.0976, 0.6615, 0.5759, 0.    , 0.3308],
       [1.2289, 1.4142, 0.9923, 0.8781, 0.3308, 0.    ]])
```

Figure 6: Euclidean distance

4.2 b)

The Goodall distances between cows is shown in Fig.7

```
array([[0.    , 0.7037, 0.7037, 1.    , 1.    , 1.    ],
       [0.7037, 0.    , 1.    , 0.75  , 1.    , 0.4537],
       [0.7037, 1.    , 0.    , 0.75  , 0.75  , 0.7037],
       [1.    , 0.75  , 0.75  , 0.    , 0.4537, 0.75  ],
       [1.    , 1.    , 0.75  , 0.4537, 0.    , 1.    ],
       [1.    , 0.4537, 0.7037, 0.75  , 1.    , 0.    ]])
```

Figure 7: Goodall distances

```
#construct dict
dic={"Ayrshir": 1/2,"Holstein": 1/3,"Finncattle": 1/6,
        "lively":1/6,"kind":1/3,"calm":1/2,
        "rock":1/3,"country":1/3,"classical":1/3}

#Get categorical features
ccols=["race","character","music"]
cdata=np.array(cows[ccols])
cdata

#Compute categorical similarity
def get_catsim_M(cdata,dic):
    lens,f_num=cdata.shape

    CM=np.zeros((lens,lens))

    for i in range(lens):
        j=i+1
        while j <lens:
            tep=0
            for x in range(f_num):
                if cdata[i][x]==cdata[j][x]:
                    tep=tep+1-dic[cdata[i][x]]**2
            CM[i][j]=CM[j][i]=1-tep/f_num
            j=j+1
    return CM.round(4)

re=get_catsim_M(cdata,dic)

re
```

4.3 c)

Distance using both numerical and categorical features is shown in Fif.8

```
array([[0.   , 1.507, 1.735, 2.1  , 2.658, 2.941],
       [1.507, 0.   , 2.253, 1.838, 2.814, 2.167],
       [1.735, 2.253, 0.   , 1.632, 1.953, 2.194],
       [2.1  , 1.838, 1.632, 0.   , 1.352, 2.164],
       [2.658, 2.814, 1.953, 1.352, 0.   , 2.069],
       [2.941, 2.167, 2.194, 2.164, 2.069, 0.   ]])
```

Figure 8: Combined distance measure

```
numsim=get_numsim_M(ndata)
catsim=get_catsim_M(cdata,dic)

a=2/5

mixSim=a*numsim/np.std(numsim)+(1-a)*catsim/np.std(catsim)
mixSim=mixSim.round(3)
mixSim
```

4.4 d)

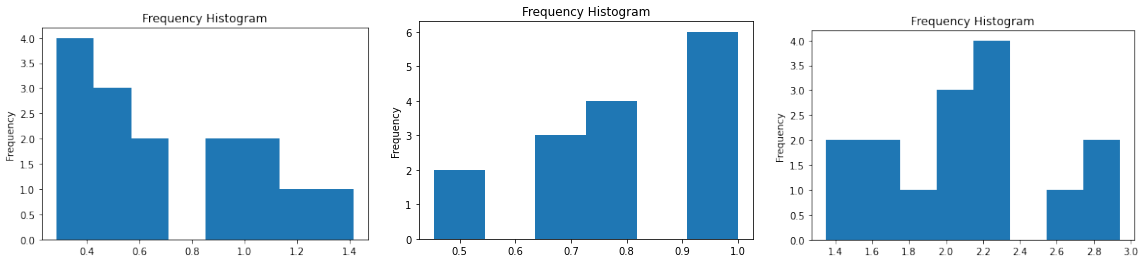
4.4.1 Histograms

```
#Show Histograms
def show_hist(data,bins):
    x =[]
    for i in range(6):
        x.extend(data[i][i+1:6])
    # print(x)
    plt.hist(x, bins=bins)
    plt.gca().set(title='Frequency Histogram', ylabel='
       Frequency')
    plt.show()
```

After test different numbers of bins in all three cases, I choose 8,6,8 in each plot as their number of bins to illustrate the clusters of cows in data. The figures are shown in Fig.9 I think the measurement combined numerical and categorical features can better cluster cows.Because more features are used for calculation, it means that we have considered more comprehensively, resulting better effect.

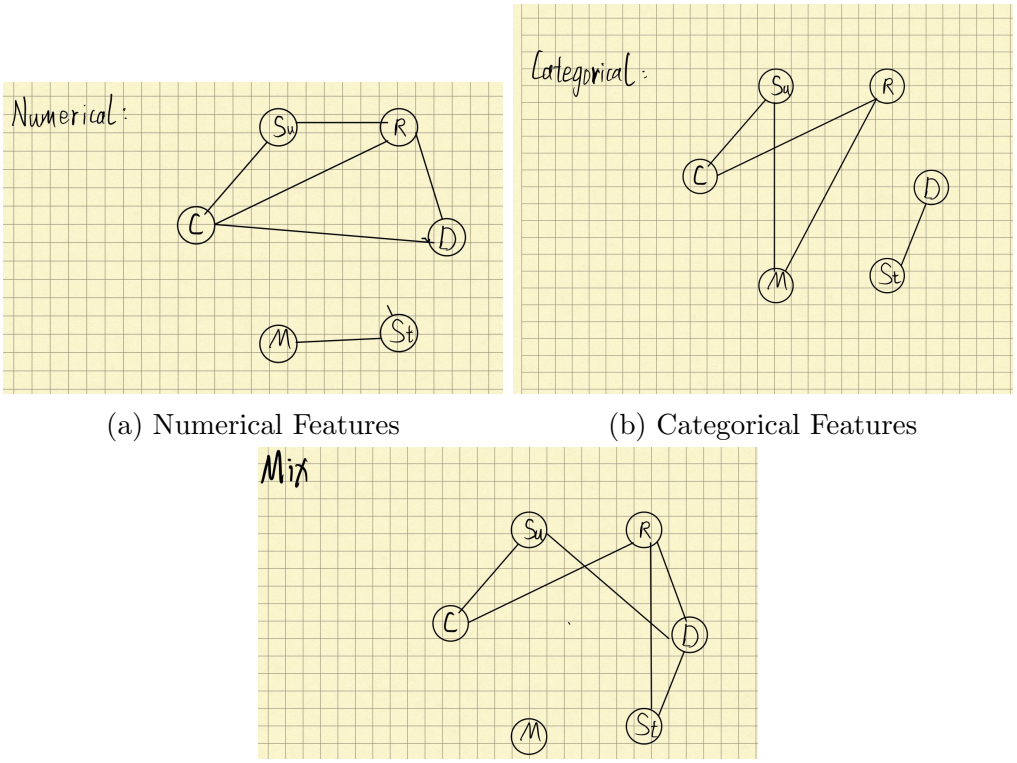
4.4.2 Graph-based clustering

First we can create a complete distance graph between cows. and we know the distances of cows from the task above. Then We should remove edges with longest distances(largest



(a) Numerical Features, Bins:8 (b) Categorical Features, Bins:6 (c) Numerical+Categorical Features, Bins:8

Figure 9: Histograms in 3 different cases



(a) Numerical Features (b) Categorical Features

(c) Numerical+Categorical Features

Figure 10: Graph clustering in 3 different cases

weights) until the graph is broken into two connected components. These components are clusters of cows.I think the measurement of mix features including numerical and categorical can better cluster cows from Fig.10.The results are not so radical to make the number of cows out of balance among the different clusters. It balances the application of two different types of data into the similarity, the more data types are used in the results, making the results more inclusive.