CS422 - Homework #1

By: Laura Pereda

In [1]:

```
import numpy as np
import pandas as pd

from sklearn import datasets
from sklearn.impute import SimpleImputer

import seaborn as sns

import matplotlib.pyplot as plt
%matplotlib inline
```

Recitation Problems

Chapter 1:

Exercise 1 - Discuss whether each of the following are data mining tasks:

- A. No, simply divinding customers based on a characteristic is not data mining. It's just querying.
- B. No. Again, dividing customers based on a characteristic isn't data mining, but querying.
- C. No, computing sales seems more like accounting than data mining.
- D. No, this follows more towards querying in a specific order than data mining.
- E. No, since the dice is fair data mining would not be useful here. Probability analysis or statistics seems more like it.
- F. Yes, creating models based on previous data follows along the lines of a data mining task.
- G. Yes, a model could be created to follow normal heart rate and use past info to determine abnormal heart rates.
- H. Yes, creating models on past seismic waves associated with earthquakes.
- I. No, this is just extraction, which is not enough to make this a data mining task.

Chapter 2:

Exercise 2 - Classifying attributes as binary, discrete, or continuous. In addition, classify as either qualitative (nominal or ordinal) or quantitative (interval or ratio).

- A. Binary; Quantitative, Ordinal
- B. Continuous; Quantitative, Ratio
- C. Discrete; Qualitative, Ordinal Ex. You can't really cut up people's opinions into parts nor does a unit of measurement exist to measure them. Most simple responses would fall under "good, better, best". Just enough to have some order.
- D. Continuous; Quantitative, Ratio
- E. Discrete; Qualitative, Ordinal
- F. Continuous; Quantitative, Interval Ex. The height above sea lvl is always changing but it can be measured.
- G. Discrete; Quantitative, Ratio
- H. Discrete; Qualitative, Nominal
- I. Discrete; Qualitative, Ordinal
- J. Discrete; Qualitative, Ordinal
- K. Continuous; Quantitative, Interval Ex. Distance always changing, can be measure, not sure about dividing/multiplying
- L. Discrete; Quantitative, Interval Ex. Density is within a measurement.
- M. Discrete; Qualitative, Nominal

Exercise 7 - Which of the following quantities is likely to show more temporal autocorrelation: daily rainfall or daily temperature? Why?

```
Tricky question, but seeing as I live within an area that temperature is categorized within
four seasons, with each season having its own set of temperature changes, I would say daily
temperature best represents temporal autocorrelation. Daily rainfall can be inconsistent th
roughout some seasons, sometimes not even showing at all. The location of the area also det
ermines the amount of rainfall. Daily temperature just seems more reliable.
```

Exercise 15 - Given a set of *m* objects that is divided into *k* groups of size m i. If the goal is to obtain a sample of size n < m, what is the difference between the following two sampling schemes? Assume sampling with replacement

The first scheme represents **stratified sampling** where there are prespecified groups of objects and we take an equal number of objects from each group. There is a variation in where we take a number proportional to the size of that particular group. The second scheme represents a simple random sampling where each object has the same probability of being selected, especially since we assume the sampling is with replacement. Sampling only works if it is a good representation of the original data set. While simple random sampling with replacement makes it easier to analyze because the probability remains the same for each object, if these m objects are different types that it will fail to represent those types of objects that are less frequent. Stratified sampling is able to cover that.

Exercise 16 - Talks about the inverse document frequency transformation.

- A. What is the effect of this transformation if a term occurs in one document? In every document?

  If the term appears in one document then it will have the maximum weight of 1, which would indicate the term uniquness. If the term appears in every document, it would have the lowest weight of 0, which would indicate its lack of uniqueness.

- B. What might be the purpose of this transformation?

  In information retrieval, eliminating the "noise" as used in data mining is very important to create efficiency and improve the time to retrieve results. This transformation will help with eliminating stop words, i.e the, a, in, is, etc.

Exercise 17 - Apply square root transformation to a ratio attribute x to obtain a new attribute x *. As part of analysis, you identify an interval (a, b) in which x* has a linear relationship to another attribute y.

- A. What is the corresponding interval (a, b) in terms of x?

  (a^2, b^2)
- B. Give an equation that relates y to x.

  y = x^2

Exercise 18 - Compares and contrast some similarity and distance measures.

- A. x = 0101010001 & y = 0100011000

  Hamming distance = 3 Jaccard similarity = f(11)/ (f(10) + f(10) + f(11)) = 2/5

- B. Which approach (Jaccard or Hamming distance) is similar to Simple Matching Coefficient and which is more simiar to cosine measure?

  The Hamming distance is similar to the SMC. The SMC is defined as "the number of matching attribute values"/"number of attributes", which can even be seen as Hamming distance/length = 1 - SMC. In terms of the Jaccard Similarity, it is similar to the Cosine Similarity because they both ignore 0-0 matches.

- C. There are 2 animals from two different species and we wish to see which genes they have in common.

  Since we are looking to see the common shared genes between the two animals, it would be more useful to use the Jaccard Similarity.
- D. Compare the genetic makeup of the same species.

  Since we are comparing humans (99% of the same genes), it would be better to use the Hamming distance which prioritizes the difference.

Exercise 19 - For vectors x and y, calculate the indicated similarity or distance measures.

- A. x = (1, 1, 1, 1), y = (2, 2, 2, 2). Cosine, correlation, Euclidean
  - Cos(x, y) = 1
  - Corr(x, y) = 0/0; undefined
  - Euclidean(x, y) = 2
- B. x = (0, 1, 0, 1), y = (0, 1, 0, 1). Cosine, correlation, Euclidean, and Jaccard
  - Cos(x,y) = 0
  - Corr(x,y) = -1

- Euclidean(x,y) = 2
- Jaccard(x, y) = 0

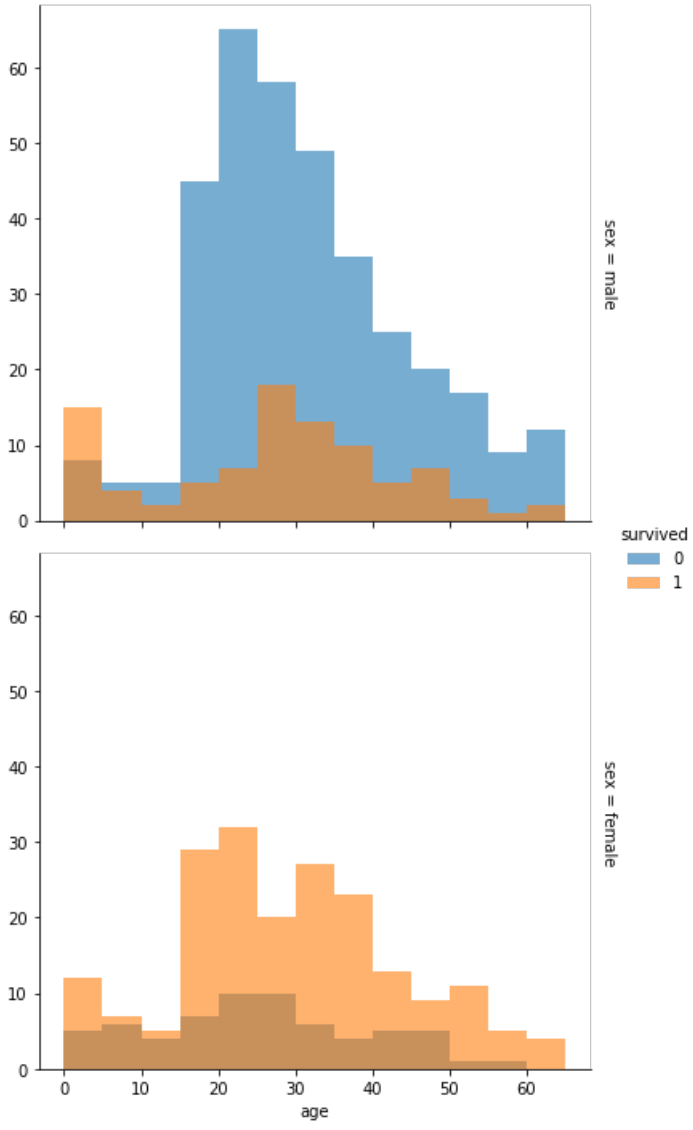
- C. x = (0, -1, 0, 1), y = (1, 0, -1, 0). Cosine, correlation, Euclidean
  - Cos(x, y) = 0
  - Corr(x, y) = 0
  - Euclidean(x, y) = 2
- D. x = (1, 1, 0, 1, 0, 1), y = (1, 1, 1, 0, 0, 1). Cosine, correlation, Jaccard
  - Cos(x,y) = 0.75
  - Corr(x,y) = 0.25
  - Jaccard(x, y) = 0.6
- E. x = (2, -1, 0, 2, 0, -3), y = (-1, 1, -1, 0, 0, -1). Cosine, Correlation
  - Cos(x,y) = 0
  - Corr(x,y) = 0

Practicum Problems

Problem 1 - Titanic Database

In [4]:

```
titanic_ds = sns.load_dataset("titanic")
bins = np.arange(0, 70, 5)
g = sns.FacetGrid(titanic_ds, row="sex", hue="survived", margin_titles=True, height=5, aspect=1.1)
g.map(sns.distplot, 'age', kde=False, bins=bins, hist_kws=dict(alpha=0.6))
g.add_legend()
plt.show()
```



From above, the relationship between age and gender shows more women survived vs men. For a man to survive, he would had to have been below the age of 10. In this case, the results do make sense since within the evacuation procedure, women and children were prioritized before men. As unfortunate as that it, the result from this histogram follow along those lines.

Problem 2 - Auto MPG Dataset

In [4]:

```
import os

autompg_ds = pd.read_csv('auto-mpg.csv', na_values=["?"])
autompg_ds = autompg_ds.drop(columns='name')
```

Demonstrate variance with missing values in horsepower:

In [5]:

```
autompg_ds.var()
```

Out[5]:

```
mpg                 61.089611
cylinders            2.893415
displacement     10872.199152
horsepower        1481.569393
weight          717140.990526
acceleration         7.604848
model_year          13.672443
origin               0.643292
dtype: float64
```

Demonstrate variance with mean replacing the missing values in horsepower:

In [6]:

```
imp_mean = SimpleImputer(missing_values=np.nan, strategy='mean')
imp_mean.fit(autompg_ds)
autompg_ds[autompg_ds.columns] = imp_mean.fit_transform(autompg_ds)
# autompg_ds['horsepower'].values.tolist() //Check to see if NaN was replaced
```

In [115]:

```
autompg_ds.var()
```

Out[115]:

```
mpg                 61.089611
cylinders            2.893415
displacement     10872.199152
horsepower        1459.177916
weight          717140.990526
acceleration         7.604848
model_year          13.672443
origin               0.643292
dtype: float64
```

Demonstrate variance with median replacing the missing values in horsepower:

In [8]:

```
autompg_ds2 = pd.read_csv('auto-mpg.csv', na_values=["?"])
autompg_ds2 = autompg_ds2.drop(columns='name')
imp_mean = SimpleImputer(missing_values=np.nan, strategy='median')
imp_mean.fit(autompg_ds2)
autompg_ds2[autompg_ds2.columns] = imp_mean.fit_transform(autompg_ds2)
# autompg_ds2['horsepower'].values.tolist() //Check to see if NaN was replaced
```

In [9]:

```
autompg_ds2.var()
```

Out[9]:

```
mpg                 61.089611
cylinders            2.893415
displacement     10872.199152
horsepower        1460.969052
weight          717140.990526
acceleration         7.604848
model_year          13.672443
origin               0.643292
dtype: float64
```

Demonstrate variance with mode replacing the missing values in horsepower:

In [10]:

```
autompg_ds3 = pd.read_csv('auto-mpg.csv', na_values=["?"])
autompg_ds3 = autompg_ds3.drop(columns='name')
imp_mean = SimpleImputer(missing_values=np.nan, strategy='median')
imp_mean.fit(autompg_ds3)
autompg_ds2[autompg_ds3.columns] = imp_mean.fit_transform(autompg_ds3)
# autompg_ds2['horsepower'].values.tolist() //Check to see if NaN was replaced
```

In [11]:

```
autompg_ds3.var()
```

Out[11]:

```
mpg                 61.089611
cylinders            2.893415
displacement     10872.199152
horsepower        1481.569393
weight          717140.990526
acceleration         7.604848
model_year          13.672443
origin               0.643292
dtype: float64
```

- Original: 1481.569393
- Mean: 1459.177916
- Median: 1460.969052
- Mode: 1481.569393

The mean imputation resulted in the lowest variance. SThis is because we are replacing the missing values with the mean. As the data becomes to take on the average, the variability of the data will decrease to 0 and thus not capture the actual data. On the otherhand, it seems the mode was able to maintain the original variance of the data, thus I would utilize the mode more often to replace any missing data.

Problem 3 - Iris Dataset

In [64]:

```
iris = datasets.load_iris()
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)

from sklearn import decomposition
pca = decomposition.PCA(n_components=3)
pca.fit(iris_df.iloc[:,0:4])
pca.explained_variance_
```

Out[64]:

```
array([4.22824171, 0.24267075, 0.0782095 ])
```

In [60]:

```
pca.explained_variance_ratio_
```

Out[60]:

```
array([0.92461872, 0.05306648, 0.01710261])
```

In [61]:

```
iris_df.var()
```

Out[61]:

```
SPL    0.685694
SPW    0.189979
PTL    3.116278
PTW    0.581006
dtype: float64
```

Calculate the percentage of variance to the original features:

In [62]:

```
total_var =  0.685694 + 0.189979 + 3.116278 + 0.581006
first_var = (0.685694/total_var)*100
second_var = (0.189979/total_var)*100
third_var = (3.116278/total_var)*100
fourth_var = (0.581006/total_var)*100

print(first_var, second_var, third_var, fourth_var)
```

```
14.994542918291165 4.154401626781096 68.14579712864128 12.705258326286472
```

The percentage of variance for the PCA components are vastly different from the percentage of variance for the original features. For example, the first feature has a variance of 14%, but the first PCA has a variance of 92%. My understanding of PCA is still a little blurry, but from what I have researched it seems that PCA helps creating new attributes out of old attributes but highlights the ones that show as much variation as possible. So, reduce data but still show equal or more information about the original data set. In this case, the first PCA was able to produce a variance of 92% which is more than any of the original features could produce.

Problem 4 - Plot Projection of each feature onto the 1st PCA from #3

In [63]:

```
#Creating new pca
pca1 = decomposition.PCA(n_components=3)
pca_ds = pca1.fit_transform(iris_df)
pca_df = pd.DataFrame(data=pca_ds, columns=['PC1', 'PC2', 'PC3'])
pca_df.head()
```

Out[63]:

	PC1	PC2	PC3
0	-2.684126	0.319397	-0.027915
1	-2.714142	-0.177001	-0.210464
2	-2.888991	-0.144949	0.017900
3	-2.745343	-0.318299	0.031559
4	-2.728717	0.326755	0.090079

In [68]:

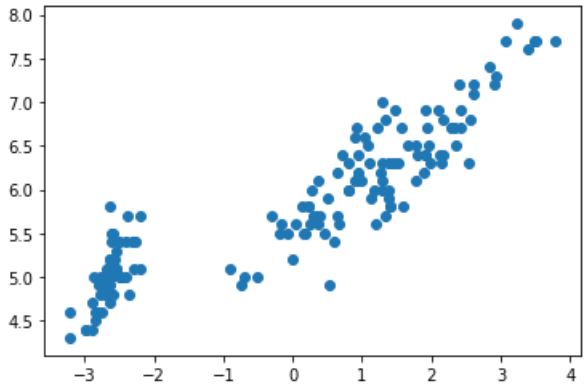
```
print("Plotting the PC1 against the first feature (sepal length in cm):")
```

```
plt.scatter(pca_df['PC1'], iris_df['sepal length (cm)'])
```

Plotting the first PCA against the first feature (sepal length in cm):

Out[68]:

<matplotlib.collections.PathCollection at 0x227bcb6e7c8>



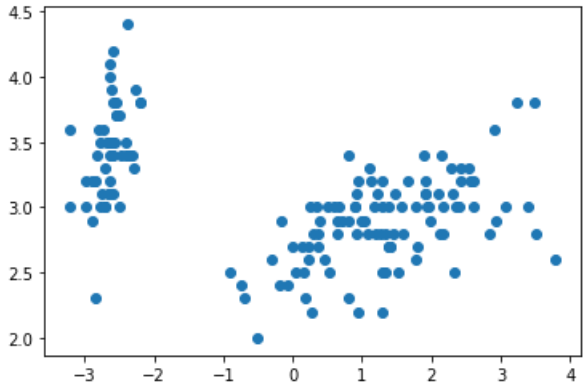
In [72]:

```
print("Plotting the PC1 against the second feature (sepal width in cm):")
plt.scatter(pca_df['PC1'], iris_df['sepal width (cm)'])
```

Plotting the PC1 against the second feature (sepal width in cm):

Out[72]:

<matplotlib.collections.PathCollection at 0x227bca15b08>



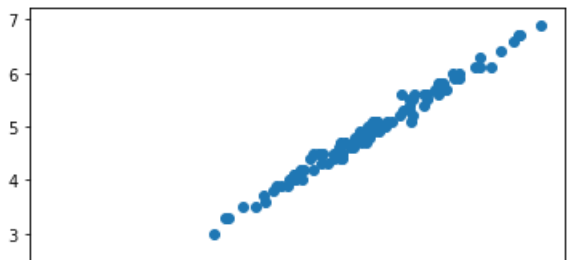
In [76]:

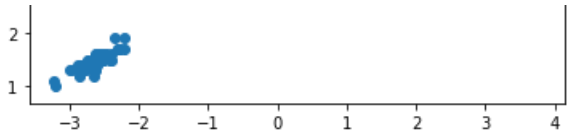
```
print("Plotting the PC1 against the third feature ():")
plt.scatter(pca_df['PC1'], iris_df['petal length (cm)'])
```

Plotting the PC1 against the third feature ():

Out[76]:

<matplotlib.collections.PathCollection at 0x227bc7ff388>





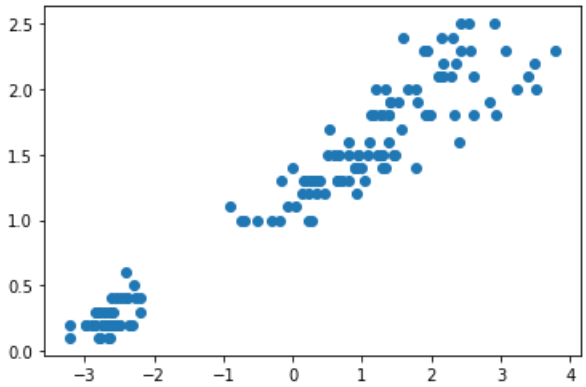
In [75]:

```
print("Plotting the PC1 against the fourth feature ():")
plt.scatter(pca_df['PC1'], iris_df['petal width (cm)'])
```

```
Plotting the PC1 against the fourth feature ():
```

Out[75]:

```
<matplotlib.collections.PathCollection at 0x227bc8478c8>
```



The scatter plots shown above show that the third feature (petal length in cm) has the closest relationship with PC1. Looking back at Problem 3, the percentage of variance for PC1 was 92% while the highest percentage of variance among the original features was 68%, which resulted in being the third feature.

Calculating the correlation coefficient between PC1 and third feature:

In [88]:

```
print(pca_df['PC1'].corr(iris_df['petal length (cm)']))
```

```
0.9978739422413102
```

Since the correlation coefficient between PC1 and third feature is almost close to 1, then I can agree with the visual inspection above.

Problem 5

Calculate the total variance for original features:

In [89]:

```
print(total_var)
```

```
4.572957
```

Calculate the total variance for the four eigen vectors (PCA):

In [90]:

```
pca = decomposition.PCA(n_components=4)
pca.fit(iris_df.iloc[:,0:4])
pca.explained_variance_
```

Out[90]:

```
array([4.22824171, 0.24267075, 0.0782095 , 0.02383509])
```

In [91]:

```
total_pca_var = 4.22824171 + 0.24267075 + 0.0782095 + 0.02383509
print(total_pca_var)
```

```
4.57295705
```

Since they seem to almost be equal, it is clear that the PCA was capable of capturing the variance of the original features and is thus a good representation with fewer features.

In terms of capturing at least 95% variance of the original data:

In [92]:

```
pca.explained_variance_ratio_
```

Out[92]:

```
array([0.92461872, 0.05306648, 0.01710261, 0.00521218])
```

The above demonstrates that the first two components are capable of capturing at least 95% variance of the original data.

In [93]:

```
first_comp = 0.92461872
second_comp = 0.05306648
print(first_comp + second_comp)
```

```
0.9776852
```

As I said before, it goes to show that PCA was capable of capturing the variance of the original features by reducing the original features to two.